Domain adaptation of word embeddings through the exploitation of in-domain corpora and knowledge bases

Hicham El Boukkouri

▶ **To cite this version:**

Hicham El Boukkouri. Domain adaptation of word embeddings through the exploitation of in-domain corpora and knowledge bases. Artificial Intelligence [cs.AI]. Université Paris-Saclay, 2021. English. NNT : 2021UPASG086 . tel-03560502

HAL Id: tel-03560502
https://theses.hal.science/tel-03560502

Submitted on 7 Feb 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Thèse de doctorat

NNT: 2021UPASG086

université PARIS-SACLAY

Domain Adaptation of Word Embeddings Through the Exploitation of In-domain Corpora and Knowledge Bases

Adaptation au domaine de plongements lexicaux via l'exploitation de corpus et de bases de connaissances spécialisés

Thèse de doctorat de l'université Paris-Saclay

École doctorale n° 580, sciences et technologies de l'information et de la communication (STIC)
Spécialité de doctorat: informatique
Unité de recherche: Université Paris-Saclay, CNRS, Laboratoire Interdisciplinaire des Sciences du Numérique, 91405, Orsay, France
Référent: Faculté des sciences d'Orsay

Thèse présentée et soutenue à Paris-Saclay, le 18 novembre 2021 par

Hicham EL BOUKKOURI

Composition du jury

François YVON Directeur de recherche, Université Paris-Saclay, CNRS, LISN	Président
Danushka BOLLEGALA Professeur, University of Liverpool	Rapporteur & Examinateur
Benoît SAGOT Directeur de recherche, INRIA Paris	Rapporteur & Examinateur
Laure SOULIER Maîtresse de conférences, Sorbonne Université	Examinatrice
Nathalie CAMELIN Maîtresse de conférences, Le Mans Université	Examinatrice

Direction de la thèse

Pierre ZWEIGENBAUM Directeur de recherche, Université Paris-Saclay, CNRS, LISN	Directeur de thèse
Olivier FERRET Expert senior, Université Paris-Saclay, CEA List	Co-encadrant
Thomas LAVERGNE Maître de conférences, Université Paris-Saclay, CNRS, LISN	Co-encadrant

Remerciements

La thèse est souvent vue comme un exercice difficile: pendant plusieurs années, on est livré à soi-même, avec plusieurs questions à explorer et souvent trop peu de réponses à apporter. Ceci étant dit, ce n'est pas le souvenir que j'en garderai. En effet, pour moi, ces années de thèse resteront avant tout des moments agréables de partage aussi bien au niveau intellectuel qu'au niveau humain. Et ce n'est pas un hasard, je dirai même que c'est bien naturel étant donné le grand soutient que j'ai reçu tout au long de ce périple.

Je souhaiterais tout d'abord remercier ma famille, bien évidemment, qui m'a soutenu durant cet exercice, mais dont la contribution s'étend en réalité bien au-delà de ces trois années de thèse. Depuis toujours, vous avez été une présence irremplaçable et sans laquelle je ne serais sûrement pas parvenu aux choses que j'ai pu accomplir aujourd'hui. Merci Malika, Driss et Othman, je vous suis éternellement reconnaissant. Je voudrais également remercier une personne spéciale qui partage et embellit ma vie au quotidien. Sophie, tu me supportes (dans tous les sens du terme) tous les jours et sans jamais rechigner. Merci d'être là et merci pour tout ce que tu fais pour moi.

Il est clair que la famille joue un rôle important lorsqu'on s'engage dans une expérience comme la thèse. Heureusement pour moi, j'ai pu voir mon cercle familial s'agrandir au fil des années grâce aux nombreuses rencontres que j'ai faites au laboratoire. Merci à tous d'avoir rendu mon quotidien agréable. Un merci plus précisément, et sans ordre particulier, à Swen, Lucie, Yuming, Sanjay, Zheng, Arnaud, Julien, Léon, Syrielle, Corentin, Juan, Mathilde, tous les derniers nouveaux ainsi que plein d'autres personnes que je ne mentionne pas mais qui sauront se reconnaître. Un merci également aux permanents, pour votre constante bienveillance, toutes les discussions intéressantes qu'on a pu avoir ainsi que tous vos précieux conseils. Merci Anne, Patrick, Gabriel, Cyril, Aurélie, Michael, Sahar, Annelies, François ainsi qu'à plein d'autres personnes encore.

Je voudrais également remercier les personnes avec lesquelles j'ai le plus échangé durant cette thèse, à savoir mes encadrants. On dit souvent que le plus important dans une thèse c'est l'encadrement de celle-ci, et je ne pourrais pas plus être d'accord. Vous avez systématiquement su me soutenir et me diriger vers la bonne

voie sans pour autant m'imposer votre vision, le tout dans une ambiance bienveillante et chaleureuse. Sans vous, je n'aurais pas été capable de mener à bien ce projet. Merci Pierre, Thomas et Olivier. J'ai commencé cette thèse en n'étant pas vraiment sûr de ce que je voulais et je la finis en n'ayant absolument aucun regret.

Enfin, je souhaiterais remercier Junichi Tsujii, Hiroya Takamura et Hiroshi Noji de m'avoir accueilli lors de mon stage au Japon. Je remercie également les organismes suivants d'avoir fourni des ressources materielles qui ont permis le bon déroulement de cette thèse: l'IDRIS, au travers de l'allocation de ressources 2021-AD011011698/R1 attribuée par GENCI, ainsi que la plateforme de calcul Lab-IA[1] pour avoir fourni des ressources de calcul; l'AI Bridging Cloud Infrastructure (ABCI)[2] pour les ressources fournies lors de mon échange au National Institute of Advanced Industrial Science and Technology (AIST); et enfin, l'Agence Nationale de la Recherche (ANR), et plus particulièrement le projet ADDICTE (ANR-17-CE23-0001), pour le financement de ma thèse.

Merci encore à toutes et à tous. Je vous souhaite une bonne continuation et espère sincèrement que nos chemins se recroiseront de très nombreuses fois encore.

[1] https://www.universite-paris-saclay.fr/plateforme-saclay-ia
[2] https://abci.ai/

Résumé

Il existe, à la base de la plupart des systèmes de TAL, des représentations numériques qui permettent à la machine de traiter, d'interagir avec et, dans une certaine mesure, de comprendre le langage humain. Ces « plongements lexicaux » se présentent sous différentes formes mais peuvent généralement être classés en deux groupes distincts : d'une part, les plongements statiques qui apprennent et attribuent définitivement une unique représentation à chaque mot ; et d'autre part, des plongements contextuels qui, à l'inverse, apprennent à générer des représentations de mots à la volée en fonction d'un contexte courant. Dans les deux cas, l'entraînement de ces modèles nécessite une quantité importante de textes. Cela conduit souvent les praticiens du TAL à collecter et fusionner des textes provenant de sources multiples, mélangeant souvent différents styles et domaines (par exemple, des encyclopédies, des articles de presse, des articles scientifiques, etc.) afin de produire des corpus suffisamment volumineux pour pouvoir entraîner des représentations de qualité suffisante. Ces corpus dits du « domaine général » sont aujourd'hui la base sur laquelle s'entraînent la plupart des plongements lexicaux, limitant fortement leur utilisation dans des domaines plus spécifiques. En effet, les « domaines spécialisés » comme le domaine médical manifestent généralement des spécificités lexicales, sémantiques et stylistiques suffisamment notables (par exemple, l'utilisation d'acronymes et de termes techniques) pour que les plongements lexicaux généraux ne soient pas en mesure de les représenter efficacement. Dans le cadre de cette thèse, nous explorons comment différents types de ressources peuvent être exploités afin soit d'entraîner de nouveaux plongements spécialisés, soit de spécialiser davantage des représentations préexistantes.

Plus précisément, nous étudions d'abord comment des corpus de textes peuvent être utilisés à cette fin. En particulier, nous montrons que la taille du corpus ainsi que son degré de similarité avec le domaine d'intérêt jouent un rôle important dans ce processus puis proposons un moyen de tirer parti d'un petit corpus du domaine cible afin d'obtenir de meilleurs résultats dans des contextes à faibles ressources. Ensuite, nous abordons le cas des modèles de type BERT et observons que les vocabulaires généraux de ces modèles conviennent mal aux domaines spécialisés. Cependant, nous montrons des résultats indiquant que des modèles formés à l'aide

de tels vocabulaires peuvent néanmoins être comparables à des systèmes entièrement spécialisés et utilisant des vocabulaires du domaine du domaine, ce qui nous amène à la conclusion que le ré-entraînement de modèles du domaine général est une approche tout à fait efficace pour construire des systèmes spécialisés. Nous proposons également CharacterBERT, une variante de BERT capable de produire des représentations de mots entiers en vocabulaire ouvert par l'exploitation d'une représentation de leurs caractères. Nous montrons des résultats indiquant que cette architecture conduit à une amélioration des performances dans le domaine médical tout en étant plus robuste aux fautes d'orthographe.

Enfin, nous étudions comment des ressources externes sous forme de bases de connaissances et ontologies du domaine peuvent être exploitées pour spécialiser des représentations de mots préexistantes. Dans ce cadre, nous proposons une approche simple qui consiste à construire des représentations denses de bases de connaissances puis à combiner ces « vecteurs de connaissances » avec les plongements lexicaux cibles. Nous généralisons cette approche et proposons également des Modules d'Injection de Connaissances, de petites couches neuronales permettant l'intégration de représentations de connaissances externes au sein des couches cachées de modèles à base de Transformers. Globalement, nous montrons que ces approches peuvent conduire à de meilleurs résultats. Cependant, nous avons l'intuition que ces performances finales dépendent en fin de compte de la disponibilité de connaissances pertinentes pour la tâche cible au sein des bases de connaissances considérées.

Dans l'ensemble, notre travail montre que les corpus et bases de connaissances du domaine peuvent être utilisés pour construire de meilleurs plongements lexicaux en domaine spécialisé. Enfin, afin de faciliter les recherches futures sur des sujets similaires, nous publions notre code et partageons autant que possible nos modèles pré-entraînés.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Context

It is hard to deny that Language has played a major role in enabling the progress of human civilisation. And while broadly speaking "Language" may be expressed in several ways (e.g. visual, auditory, tactile), written text has probably contributed the most to bringing societies together and enhancing our ability to communicate with each other. However, with recent technological advancements leading to the exchange of an ever-increasing amount of textual information, building systems to decode and process the rich variety of natural languages has become more important than ever before.

Natural Language Processing (NLP) is a field at the intersection of linguistics, computer science and statistics which aims to provide artificial machines with the ability to understand, interpret and leverage textual information. Early approaches to NLP were mainly rule-based and required in-domain knowledge from human experts to define task-specific rules (Mauldin, 1984; Klatt, 1987). However, as statistical learning became more and more prevalent, due to the growing availability of sufficiently large corpora, the increase in processing power as well as the adoption of methods like Maximum Entropy (MaxEnt) (Jaynes, 1957; Good, 1963), Support Vector Machines (SVM) (Cortes and Vapnik, 1995), Random Decision Forests (Ho, 1995) and Boosting (Freund and Schapire, 1997), the rigid rule-based approach was slowly abandoned in favor of a more flexible Machine Learning (ML) paradigm.

Instead of relying on pre-defined rules, ML models are able to learn from a set of examples and to generalize this knowledge to new unseen scenarios. However, this process still requires human experts to provide the model with a relevant description of all examples through special variables called *features*. In the specific context of NLP, raw textual information is unintelligible to a computer and byte

sequences cannot be directly leveraged to achieve any significant level of language comprehension[1]. As a result, this information has to be converted into numerical features—often using vectorial representations—which lend themselves to the various statistical computations occurring within a learning algorithm.

Early ML-oriented textual representations relied on the so-called Bag-of-Words approach (BoW), where syntax and word order are disregarded to focus solely on word occurrences. In this context, a text is represented as a vector of word-level heuristics, like word frequency (Luhn, 1957) or TF-IDF (Spärck Jones, 1972), which is supposed to provide a somewhat useful characterization of the text's content. However, since only a limited subset of words ever appears in any given text, BoW representations are often particularly sparse. Moreover, as the size of these vectors is equal to the total number of words, these representations also tend to be exceedingly large, leading to an important memory footprint that only grows exponentially when including the often necessary N-*gram* information—i.e. additional statistics about contiguous sequences of N words. In order to mitigate some of these issues, dense representations of texts were also proposed, with methods like Latent Semantic Analysis (LSA) (Landauer and Dumais, 1997), compiling count-based features over multiple contexts (e.g. sentences, documents) before applying a Singular Value Decomposition (SVD). However, these only address the *curse of dimensionality* (Bellman, 1966) at inference-time, as constructing large matrices of text statistics is still required prior to the dimensionality reduction step.

Somewhat concurrently, a different approach emerged from the area dealing with Artificial Neural Networks (ANN), a special kind of ML algorithms that leverages an iterative optimization process called Back-Propagation (Rumelhart et al., 1986; Werbos, 1990) to automatically learn internal hierarchies of features. Motivated by prior work advocating learning *distributed representations* of symbolic concepts (Hinton et al., 1986), these neural models have been applied in the context of NLP for learning distributed representations of words, more commonly known as *word embeddings*. Initially, these word representations were trained as part of task-specific architectures, where they were shown to improve over traditional hand-crafted features (Bengio et al., 2000). However, word embeddings trained on preliminary tasks such as Language Modeling (LM)—which consists in predicting upcoming words in a sentence—were soon found to encode general semantic and syntactic information that could be effectively transferred to a variety of downstream tasks (Collobert and Weston, 2008; Collobert et al., 2011). Consequently, pre-training word embeddings to serve as out-of-the-box text encoders has become a major research direction. Today, building better word embeddings is arguably the cornerstone of most state-of-the-art NLP systems, especially for

[1]However, we will discuss in Chapter 5.3 how these byte sequences can be leveraged indirectly to improve state-of-the-art models.

applications in specialized areas such as the medical domain.

1.2 Motivation

Neural Networks have become the *de facto* framework for constructing high-performance NLP systems. Usually, model architectures consist in pre-trained word embeddings, which are often fed to a general-purpose encoder, followed by one or more task-specific layers. Generally speaking, these word embeddings are either *static*, where each word is assigned a single dense representation (e.g. WORD2VEC, Mikolov et al. (2013)); or *contextual*, where the same word may have a different representation depending on its context (e.g. ELMO, Peters et al. (2018a)). In practice, word embeddings are rarely trained from scratch on every instance of a new task, but rather *pre-trained* once, then transferred onto various downstream tasks. The go-to task for this pre-training is often a variant of Language Modeling as this allows to leverage large amounts of unlabeled texts. As a result, and since the size of these pre-training corpora is known to positively correlate with the performance of learned representations, many texts from various sources are often pooled together (e.g. news, online encyclopedia, novels) into what is often referred to as *general-domain* corpora.

General-domain corpora are often large, easily available and yield satisfactory word representations. However, when the target task involves more specialized kinds of texts, such as legal contracts, clinical notes or financial reports, these general-domain representations become less relevant and lag behind word embeddings trained specifically for these domains. Moreover, it is possible to argue that any given task—as general as it may seem—represents its own specific sub-domain, and that, therefore, being able to build tailored word representations, or further specialize existing ones, is beneficial in a broader context than just providing better embeddings for specialized domains. In this dissertation, and with the medical domain and the English language as a recurring use-case, we explore how such solutions can be constructed by leveraging two kinds of in-domain resources, namely: corpora and knowledge bases.

1.3 Research Objectives

Clearly put, our goal is the following:

> *Given a target specialized domain, improve the quality of general-domain word representations using in-domain corpora and/or knowledge bases.*

In the following paragraphs, we provide useful information relative to specific concepts that are involved in this objective formulation:

Target Specialized Domain A randomly-chosen specialized domain may not have access to both in-domain corpora and knowledge bases. In order to be able to experiment with either source of in-domain knowledge, as well as to combine methods leveraging both resources at the same time, we frame our analysis in the specific setting of the medical domain and the English language. In this context, we are able to access large corpora such as MIMIC-III[2] (clinical domain) and PubMed[3] (biomedical domain), as well as various terminologies and ontologies from the UMLS[4] Metathesaurus.

Word Representations There are a number of word embedding methods as well as a variety of pre-trained representations that can be used out-of-the-box. In this thesis, we consider both static and contextual types of word embeddings and rely on, whenever appropriate, publicly available pre-trained models.

Embedding Evaluation Improving word embeddings assumes the ability to measure the quality of a given set of word representations, which is not at all trivial. In the context of this thesis, however, we suppose that the downstream performance of an embedding indirectly reflects its quality. As a result, if an embedding A, used in conjunction with some task-specific architecture, improves the performance on a given task over some other embedding B, then A is considered to be "better" in this specific context. Generalizing this idea, if an embedding A systematically improves over some embedding B on several evaluation tasks, then we consider that, overall, "embedding A is better than embedding B". Moreover, in order to accurately compare different systems, we systematically perform multiple random restarts and use the resulting performance distribution to compute a mean score, a standard deviation as well as, for later experiments, carry out statistical significance tests. The topic of embedding evaluation is discussed in more detail in Chapter 3.

1.4 Contributions

Along this thesis, we make the following contributions:

- We study how both the corpus size and degree of specialization affect downstream performance in a specialized domain. We validate the intuition that larger corpora produce better representations but also show that large general corpora may perform better than small specialized ones. Moreover, we

[2] https://physionet.org/content/mimiciii-demo/1.4/
[3] https://www.nlm.nih.gov/bsd/pmresources.html
[4] https://www.nlm.nih.gov/research/umls/index.html

also show evidence that the degree of specialization becomes more important than additional size when dealing with specialized corpora that are already sufficiently large.

- We propose a method for specializing general-domain embeddings in a low-resource context. More specifically, we consider a worst-case scenario where only the target task corpus is available and carry out the following procedure: first, we train static representations on the task corpus, then, we resume the pre-training of general-domain contextual embeddings on the same task corpus, and finally, we combine both static and contextual representations into one final model. Despite only relying on resources that are readily available for most domains (i.e. a target task corpus and pre-trained contextual embeddings from the general domain), we show that this method consistently improves over only using general pre-trained embeddings out-of-the-box.

- In the specific context of BERT-like (Devlin et al., 2019) models that rely on a vocabulary of subwords for handling unknown tokens, we tackle the issue of using a general-domain vocabulary in a specialized domain. Specifically, since most specialized versions of these models merely re-train them on a specialized corpus all the while keeping the original general-domain vocabulary, we explore the impact this may have, beginning with the quality of tokenization which we show to be worse than what is produced by a domain-specific subword vocabulary.

- We also evaluate the usual re-training strategy against similar models which are trained from scratch on specialized corpora using a specialized vocabulary. We show that the latter configuration generally outperforms models re-trained from a general-domain version, however, we also note that this difference is usually relatively small. Setting aside possible issues related to the pre-training procedure (e.g. insufficient training), and given the availability of general-domain models which can be used to initialize training for any specialized domain, we conclude that re-training from a general model is still appropriate as it is less expensive and leads to comparable, although slightly lower, performance.

- Despite the fact that specialized models using general-domain subword vocabularies may perform on par with fully specialized ones, we argue that having a subword-based tokenization system is still inconvenient in practice (e.g. splitting a single word into multiple subwords). In an effort to go back to simpler word-level models, we propose CharacterBERT, a variant of BERT that does not rely on subwords but rather consults word characters to produce word-level contextual representations. We conduct an evaluation on

several medical tasks and show that overall, this model outperforms BERT versions that have been in identical conditions. Moreover, we provide evidence supporting the fact that CharacterBERT is more robust to noisy texts containing several types of misspellings.

- We explore ways to specialize general-domain representations using knowledge bases. More specifically, we develop a strong baseline using a simple method relying on graph embeddings and concatenation. In this framework, we restrict the knowledge base to the single most common type of relation `is_a`, which produces a single graph. Then, we embed this knowledge graph using graph embedding methods[5]. Finally, we combine the word and knowledge embeddings using a simple concatenation to produce a final set of representations. We conduct an extensive evaluation here as well and demonstrate that both static and contextual embeddings may effectively be specialized using this simple approach.

- We extend the previous method to the specific case of BERT-like models by proposing new layers called Knowledge Injection Modules (KIM) that inject the knowledge representations directly within the model architecture. Following the standard re-training approach (i.e. general domain then specialized corpora), we pre-train medical versions of BERT and CharacterBERT as well as versions relying on Knowledge Injection Modules, then evaluate these models on several medical tasks. Our results indicate that pre-training on a specialized corpus using KIMs is indeed able to improve performance over plain re-training with a standard architecture. However, we note that these results vary from one evaluation task to another, possibly suggesting that this approach may be more relevant in some situations than others.

Overall, our contributions tackle rather independent facets of the main issue. As a result, we expect that some of these methods could be used together to achieve an even greater degree of specialization of general-domain word representations. Nevertheless, it is also fair to point out that our experiments focused on a single setting (i.e. the medical domain and the English language) and may very well not generalize perfectly to other languages and domains. However, we claim that the methodology that we propose can be applied in other settings as well, since our methods are arguably agnostic to both the language and domain parameters. Ultimately, we hope that our results hint at more general properties that would hold in other similar settings. In an effort to facilitate future research on the

[5] As opposed to knowledge base embedding methods that would generally leverage different types of relations.

adaptation of word embedding to specialized domains, we open source our code and share our pre-trained models whenever appropriate[6].

1.5 Thesis Outline

This manuscript follows a simple outline which starts with a preliminary chapter on background knowledge and related work (§2). This chapter introduces information that is relevant to specializing word representations using in-domain knowledge. The first section defines Transfer Learning and present ways in which it can be used to leverage related corpora (§2.2). More specifically, we provide an overview of Transfer Learning (§2.2.1) then proceed to focus on two specific aspects of it, namely: Sequential Transfer (§2.2.2), which we present as the main approach that is currently backing word embedding models; and Domain Adaptation (§2.2.3), which includes methods to specialize models for target domains. Then, we provide information on how to leverage external knowledge as well (§2.3). Specifically, we introduce preliminary concepts related to ontologies and knowledge bases (§2.3.1). Then, we present methods that leverage these external sources of knowledge to improve existing embeddings, both static (§2.3.2) and contextual (§2.3.3). Finally, we introduce meta-embeddings, as an additional and perhaps unorthodox way for enhancing word representations with external knowledge (§2.3.4).

This is followed by a second preliminary chapter that briefly discusses the topic of embedding evaluation (§3). Specifically, we give an overview of embedding evaluation, discussing the difference between intrinsic and extrinsic approaches (§3.2), then, we provide details about our own approach to evaluation through means like random restarts, model ensembles and statistical significance (§3.3).

The next chapters follow closely our list of contributions. First, we focus on leveraging text corpora (§4), which we show can be used to improve existing embeddings, confirming in the process the importance of both corpus size and domain similarity (§4.2). We then propose a method for leveraging a small in-domain corpus along with other widely available resources to produce results that are on par with training in-domain representations (§4.3).

The following chapter tackles BERT-like models that use a subword vocabulary (§5). In this context, we demonstrate that the original BERT vocabulary may not be suited for specialized domains and train parallel models from scratch, on corpora from different domains, using different vocabularies, to test our hypothesis (§5.2). We then propose CharacterBERT, a new variant of BERT that does not rely on a subword system and consults instead input tokens' characters to produce word-level contextual representations (§5.3).

[6] https://github.com/helboukkouri/phd-code

Subsequently, we return to the topic of enhancing word representation with in-domain information, this time focusing on using knowledge bases (§6). With simplicity in mind, we develop a strong baseline consisting in knowledge base embeddings (§6.2) and concatenation (§6.3) and demonstrate that it can lead to important improvements when applied to both static word representations as well as transformer-based models like BERT and CharacterBERT. Moreover, we propose Knowledge Injection Modules (KIM) to directly incorporate knowledge representations within the architecture of BERT-like models and show promising results which we argue, however, to depend on how relevant the chosen external knowledge is to the target task.

Finally, we conclude this manuscript (§7) by providing a summary of our findings (§7.1), going one last time over our contributions (§7.2) and proposing multiple future directions to further explore ways in which to improve word representations with in-domain knowledge (§7.3).

1.6 Publications

Some of the work that we present in this manuscript has been the object of submissions and/or publications in NLP conferences:

- Hicham El Boukkouri, Olivier Ferret, Thomas Lavergne, and Pierre Zweigenbaum. 2019. Embedding strategies for specialized domains: Application to clinical entity recognition. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 295–301, Florence, Italy. Association for Computational Linguistics

- Hicham El Boukkouri. 2020. Ré-entraîner ou entraîner soi-même ? stratégies de pré-entraînement de BERT en domaine médical (re-train or train from scratch ? pre-training strategies for BERT in the medical domain). In *Actes de la 6e conférence conjointe Journées d'Études sur la Parole (JEP, 33e édition), Traitement Automatique des Langues Naturelles (TALN, 27e édition), Rencontre des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues (RÉCITAL, 22e édition). Volume 3 : Rencontre des Étudiants Chercheurs en Informatique pour le TAL*, pages 29–42, Nancy, France. ATALA et AFCP

- Hicham El Boukkouri, Olivier Ferret, Thomas Lavergne, Hiroshi Noji, Pierre Zweigenbaum, and Jun'ichi Tsujii. 2020. CharacterBERT: Reconciling ELMo and BERT for word-level open-vocabulary representations from characters. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6903–6915, Barcelona, Spain (Online). International Committee on Computational Linguistics

- **[Under Submission]** Hicham El Boukkouri, Olivier Ferret, Thomas Lavergne, and Pierre Zweigenbaum. 2021. A Simple Approach to Specializing Both Static and Contextual Embeddings Using Knowledge Graphs.

Chapter 2

Background and Related Work

2.1 Introduction

This chapter provides background knowledge about two main ways in which word representations may be constructed and further improved, namely: leveraging related tasks and corpora, and leveraging external knowledge in the form of ontologies and knowledge bases. These two approaches are also the two main axes through which we explore the topic of this thesis, with Chapter 4 focusing on utilizing text corpora and Chapter 6 focusing more on external resources and knowledge bases. In the following, we first address leveraging related tasks and corpora within the perspective of Transfer Learning, for which we provide an overview. We also show how word embeddings can be positioned within this Transfer Learning framework as a direct application of Sequential/Model Transfer. We then provide a brief review of Domain Adaptation and show how model pre-training, which is traditionally seen as a transfer method, can be used to adapt general-domain representations or monolingual models to new domains and languages. Eventually, we address the topic of external knowledge and how it can be leveraged to improve NLP systems, more specifically word embeddings. After briefly introducing some of the main sources of external knowledge, we delve into how both traditional (i.e. static) word representations as well as more modern pre-trained language models can be improved using such knowledge. Ultimately, we discuss meta-embeddings, an approach that aims to combine multiple sets of word representations, and talk about how these aggregated embeddings can be used in a knowledge injection context to produce specialized word representations.

2.2 Leveraging Related Tasks and Corpora

Three main scenarios generally occur in Machine Learning: *unsupervised learning*, when only unlabeled data is used; *supervised learning*, when only labeled data is used; and *semi-supervised learning*, when both labeled and unlabeled data is used. In the traditional supervised learning setting, there are two inapparent assumptions which are often made. First, it is assumed that the available data is large enough to enable the training of a model with good performance. Second, it is also assumed that any future data is similar enough that the trained model will have no difficulty generalizing to unseen examples. In practice, however, these assumptions rarely hold, and most often than not the actual problems which are tackled do not come with enough labeled examples (i.e. low resource setting), and future data is frequently different enough that the performance of trained models slowly deteriorates with time (i.e. domain shift (Gretton et al., 2009)). As a result, modern NLP systems usually rely on Transfer Learning (TL), a set of tools that leverages data from different tasks and/or domains (called *source*) to build models with improved performance on a specific task and/or domain of interest (called *target*). In this context, our goal, which we recall to be specializing word representations using in-domain resources, can be seen as leveraging existing models from a source domain (i.e. general-domain embeddings), along with resources from the target domain (i.e. in-domain corpora and knowledge bases), in order to produce better word representations for this specialized domain.

2.2.1 An Overview of Transfer Learning

While most views agree on the general definition of Transfer Learning, there is nonetheless a variety of ways to categorize its different techniques (Shao et al., 2014; Tan et al., 2018). One such way that is relatively up-to-date with current state-of-the-art is (Ruder, 2019) which condenses Transfer Learning methods into two greater categories that ultimately lead to four main approaches (see Figure 2.1). On one hand, *transductive transfer* deals with situations where the source and target tasks are similar but their domains differ. This typically includes Domain Adaptation methods (DA), which deals with different domains within the same language, and Cross-lingual Learning (CL), when the two domains are actually instances of two different languages. On the other hand, *inductive transfer* tackles situations where the source and target domains may be similar but the actual tasks differ. In practice, this leads to instances of Multi-task Learning (MTL), when two or more tasks are learned in parallel, or Sequential Transfer Learning (STL) when these tasks are learned sequentially.

In what follows, we will be focusing on Sequential Transfer Learning (§2.2.2) and Domain Adaptation (§2.2.3) as these will be covering most of the methods

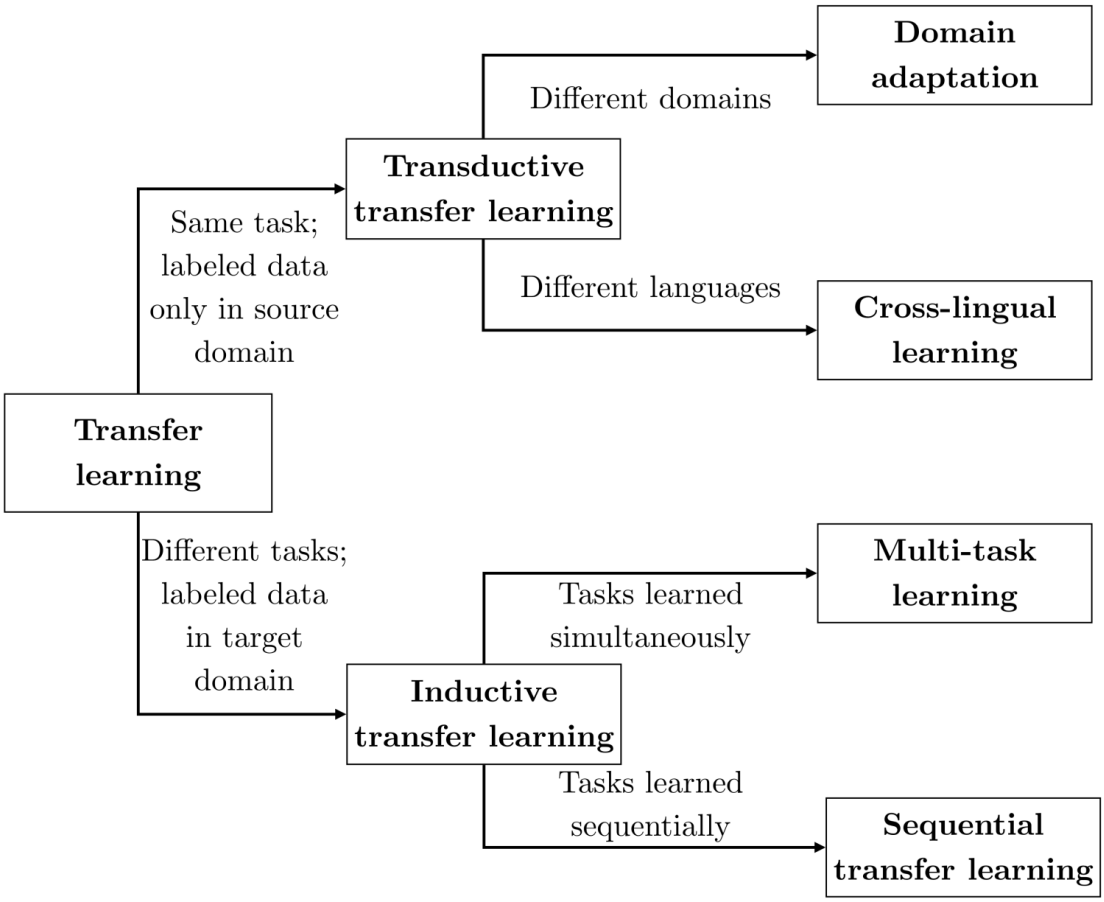

Figure 2.1: A taxonomy for transfer learning in NLP. Credits go to Ruder (2019).

presented in this manuscript. Nevertheless, it is interesting to note that while these two approaches are different in theory, they can be difficult to tell apart in practice. For example, re-training a general model on a specialized corpus may be seen as a Domain Adaptation method, however, it may very well be seen as an instance of Sequential Transfer as well since the features that were learned during the initial general-domain pre-training are used to initialize a second pre-training on the specialized corpora. Generally speaking, one useful rule of thumb is that Sequential Transfer mostly aims to learn "general features" that can be transferred over various downstream tasks and/or domains, while Domain Adaptation mostly focuses on learning "domain-specific features" that will be useful for a specific domain of interest.

2.2.2 Sequential Transfer Learning

In this section, we will be going over some early non-NLP work that leveraged Sequential Transfer before delving into the different steps of STL and how it relates to training word representations. Finally, we briefly mention recent work that helps alleviate some forgetting issues that arise when training models sequentially.

Sequential Transfer in Computer Vision

Sequential Transfer Learning, also known as Model Transfer (Wang and Zheng, 2015) or sometimes simply "Transfer Learning"[1], refers to a general approach where a preliminary model is built on a source task before being transferred to an often related but usually different target task with the aim of improving downstream performance. Originally, this approach gained traction within the Computer Vision (CV) community when models trained on large labelled datasets such as ImageNet(Deng et al., 2009; Russakovsky et al., 2015) were shown to significantly improve results on other downstream tasks (Yosinski et al., 2014; Huh et al., 2016), with notable gains on image classification (Donahue et al., 2014), image segmentation (Dai et al., 2016) and image captioning (Karpathy and Li, 2015); especially when dealing with small challenging datasets (Oquab et al., 2014). Today, state-of-the-art vision models are rarely built from scratch; instead, models pre-trained on large image datasets are re-used and adapted in order to boost performance on various target tasks (Radford et al., 2021; Goyal et al., 2021).

[1]In fact, although Transfer Learning as a whole includes other approaches like Multi-task Learning and Domain Adaptation, the term "Transfer Learning" is often mistakenly used to refer specifically to the Sequential Transfer Learning paradigm.

Sequential Transfer in Two Phases

Sequential Transfer Learning consists of two phases: a *pre-training* phase and a *transfer* phase[2]. With pre-training, a distinction can be made between the supervised and unsupervised paradigms, with supervised pre-training referring to situations where the model is pre-trained on a supervised task such as image classification or object detection on ImageNet. In the context of Natural Language Processing, there have been instances of successful model transfer using supervised pre-training. For example, neural text encoders pre-trained on Natural Language Inference datasets have been shown to benefit several types of downstream tasks such as text classification, textual entailment and paraphrase detection (Conneau et al., 2017a). However, most successful attempts to pre-train NLP systems have relied on unsupervised pre-training, which is able to leverage large amounts of unlabeled texts[3]. While some pre-training procedures are purely unsupervised, such as Latent Dirichlet Allocation (LDA) (Blei et al., 2001) which automatically learns a set of topics that describe the input texts, most of them however are *self-supervised*[4], meaning that labeled examples are generated from the unlabeled data, typically according to some reconstruction objective that aims to recover missing parts of the original input, effectively making these pre-training tasks supervised. In practice, the reconstruction task is often a variant of Language Modeling (LM) where the source model has to predict the next word in a sentence (i.e. traditional language modeling) or fill-in missing words (e.g. masked language modeling). Presumably, training on these tasks encourages the source model to acquire some level of linguistic knowledge which can then be transferred to downstream tasks resulting in improved performance. This transfer can be achieved through one of two main methods: either using the pre-trained model output as fixed features (a.k.a *feature extraction*), or by adapting the pre-trained model on the target task *via* further training—often using the model as a component within a larger task-specific architecture (a.k.a *fine-tuning*).

In what follows, we discuss Sequential Transfer Learning from a specific perspective that is closely related to the topic of this thesis: *representation learning*, and more precisely, *word embeddings* and how training and using these representations can be seen as an instance of Sequential Transfer.

[2]Sometimes, this is called the "adaptation" phase. However, we prefer the term "transfer" as it is less likely to be confused with Domain Adaptation methods.

[3]For instance, the Common Crawl Project generates several tera-bytes of text data, crawled from the internet, every month: `http://commoncrawl.org/`

[4]As recently coined by Yann LeCun.

Word Embeddings & Transfer Learning

Word embeddings are dense numerical representations of words that are learned from usually large text corpora, and serve as features for encoding raw textual information in machine learning models, particularly neural networks. As such, training and using word embeddings can be seen as an application of Sequential Transfer where the source model (i.e. the word embeddings) is transferred to improve performance on target tasks.

There are numerous word embedding methods and they all broadly fall into one of two categories: *static embeddings* and *contextual embeddings*. Static embeddings assign a single representation to each word regardless of its context. These are methods like WORD2VEC (Mikolov et al., 2013), which assigns random representations then adapt them so that words that co-occur together have similar word vectors; GLOVE (Pennington et al., 2014), which achieves a similar goal by essentially decomposing a matrix of co-occurrence statistics[5]; and FASTTEXT (Bojanowski et al., 2017), which improves over WORD2VEC by allowing unseen words to be represented according to their spelling using sub-words units. However, these static representations are unable to properly represent words with multiple meanings which vary depending on the context. This may ultimately jeopardize downstream performance as, for instance, a word like "bank" would be assigned an identical representation in a financial context (e.g. "I withdrew some money from the bank.") and a scientific context (e.g. "In geography, a bank is the land alongside a body of water."). To address this issue, some methods aim to learn multiple representations for each word, either according to a pre-defined number of senses (Chen et al., 2015), or automatically learning the different word senses on the fly (Neelakantan et al., 2014; Trask et al., 2015). These techniques, however, were soon overtaken by methods that are able to take the context into account to produce a continuous spectrum of representations, with early contextualization efforts like CONTEXT2VEC (Melamud et al., 2016), generalizing the WORD2VEC objective by learning recurrent representations of word contexts[6]; and COVE (McCann et al., 2017), pre-training a similar architecture on Machine Translation datasets. While these methods were not proposed as contextual word embeddings *per se*, they pre-train recurrent neural networks which are able to produce context-dependent word representations at each time step. Ultimately, ELMO (Peters et al., 2018a) trained an actual word embedding model with contextual representations in mind using

[5]While the original formulation of the GLOVE objective is more akin to a regression than a matrix factorization, it has been shown that this objective essentially applies an iterative matrix decomposition of co-occurrence statistics (see Section 2.4 of Levy et al. (2015)).

[6]Technically, CONTEXT2VEC does not aim to learn contextual representations of words *per se* but rather uses a recurrent model for modeling a word's context. However, this method may be seen as a precursor to modern contextual models since the recurrent context representations may be used as approximate contextual embeddings for their central words.

recurrent neural networks. The model was pre-trained using a language modeling objective on large general-domain corpora, and was successfully transferred via feature extraction to improve downstream performance on several NLP tasks.

Most of the aforementioned methods pre-train a source model using self-supervised language modeling tasks. However, it is important to note that various efforts have successfully pre-trained models in a purely supervised setting as well, namely: using paraphrases (Wieting et al., 2016), image captions (Kiela et al., 2018a) and dictionary definitions (Hill et al., 2016). Nevertheless, these pre-trained representations were usually transferred in a *feature extraction* fashion where static, contextual and sometimes both types of embeddings (Peters et al., 2017) were used as fixed components in a model that is trained from scratch on the target task. While such approaches have led to significant performance boosts on various kinds of tasks (Collobert et al., 2011; Pennington et al., 2014; Xiong et al., 2018), training task-specific models from scratch is ultimately not desirable. Moreover, static word representations, which were traditionally used as fixed features, have been shown to transfer better when fine-tuned along with the target architecture (Kim, 2014a). This has led to more models being transferred *via* fine-tuning, which has become standard when transferring pre-trained models. Nevertheless, fine-tuning large pre-trained models is not always practical either and can be costly in both computation and labeled data (Dai and Le, 2015; Mou et al., 2016).

More recently, ULMFiT (Howard and Ruder, 2018) showed that deep neural language models trained on large corpora could be successfully transferred as a whole and fine-tuned to reach state-of-the-art performance on text classification tasks. Through a number of steps including a general pre-training, a second task-specific pre-training with layer-wise optimization and a final fine-tuning with gradual layer unfreezing, ULMFiT demonstrated that model transfer with end-to-end fine-tuning was possible with only a limited number of labeled examples from the target task. In the meantime, a novel neural architecture called Transformer was proposed (Vaswani et al., 2017), improving over traditional convolutional and recurrent layers in both speed and performance. This paved the way for new methods like OpenAI's GPT (Radford et al., 2018) which proposed a unified architecture for transferring language models to a variety of NLP tasks including Natural Language Inference, Sentence Similarity and Question Answering. Soon after, the BERT model was proposed (Devlin et al., 2019), deeply impacting the way modern NLP is performed. Building on the observation that standard language modeling could only leverage unidirectional information[7], BERT pre-trained a deep architecture of Transformer encoders on a Masked Language Modeling (MLM) task, a bi-directional alternative that consists in predicting masked words

[7]Models like ELMo try to achieve bidirectionality by concatenating representations from two unidirectional models, however, this is not the same as having a strictly bidirectional architecture.

in a sentence, similar to a *cloze* task (Taylor, 1953). Moreover, BERT was trained on Next Sentence Prediction (NSP), a task that aims to determine whether two input sentences are consecutive or randomly sampled, which supposedly[8] allows the model to better handle tasks that are at the sentence-pair level, such as like Natural Language Inference. More importantly, BERT could be seamlessly fine-tuned, with a minimal amount of additional parameters and in an end-to-end fashion, to reliably achieve state-of-the-art performance on a wide variety of NLP tasks.

Since the advent of BERT, the sequential transfer of transformer-based architectures followed by an end-to-end fine-tuning has become ubiquitous; with variants incrementally improving over this original model spawning regularly (e.g. RoBERTa (Liu et al., 2019), ALBERT (Lan et al., 2020)). Notwithstanding, some of these approaches stand out for specific reasons. For instance, T5 (Raffel et al., 2019) attempts to unify various frameworks by casting NLP tasks as text-to-text problems. GPT-2 (Radford et al., 2019) and GPT-3 (Brown et al., 2020) push the training of large models on large corpora to the extreme and demonstrate interesting zero and few-shot capabilities for such large pre-trained language models. Finally, ELECTRA (Clark et al., 2020) explores pre-training models that discriminate true texts from realistic artificial inputs in an adversarial-like setting[9].

Catastrophic Forgetting & Adapter Modules

Most modern NLP pipelines rely on fine-tuning large pre-trained models, which can quickly become inefficient when dealing with multiple target tasks. In fact, while earlier language models like BERT could be reasonably fine-tuned for a single target task using modern processing units, more recent architectures tend to be significantly larger (e.g. GPT-3), making them prohibitively expensive to use and even more expensive to fine-tune on several tasks. Moreover, training such large models has been known to be unstable (Dodge et al., 2020), with a high sensitivity to optimization parameters that makes them prone to catastrophic forgetting (McCloskey and Cohen, 1989; French, 1999)—a phenomenon where a model forgets everything it has learned so far when re-trained on a new task. To deal with these issues, Adapter Modules have been proposed, first in the context of Computer Vision (Rebuffi et al., 2017) and eventually for Natural Language Processing (Houlsby et al., 2019). These small layers can be placed within the architecture of a pre-trained model like BERT such that during fine-tuning, all original model parameters are fixed and only the adapter layers are trained. As a result, the original pre-trained model is preserved and a new task-specific module

[8] In subsequent work the Next Sentence Prediction task has been shown to degrade the overall quality of the model (Yang et al., 2019; Joshi et al., 2020).

[9] The authors do not actually use an adversarial objective but instead rely on maximum likelihood.

may be learned for each new task, leveraging information from the source model. Along with *feature extraction* and end-to-end *fine-tuning*, Adapter Modules are today one of the three main ways in which pre-trained models can be leveraged to benefit downstream target tasks.

2.2.3 Domain Adaptation

In this section, we will be providing a definition of Domain Adaptation, as well as a taxonomy for its methods. Then, we will focus on methods that aim to achieve such adaptation by leveraging *data* in particular (i.e. as opposed to altering the model architecture or training objective). Specifically, we go over *data selection* and *pseudo-labeling* as two common data-centric techniques for domain adaptation. Finally, we delve into pre-training as a third approach that is especially relevant in the context of modern, language model-based, word representations.

Definition & Taxonomy

Contrary to Sequential Transfer where the goal is to learn general features that can be useful for a variety of downstream tasks, Domain Adaptation usually seeks to learn domain-specific features that are relevant to a particular target domain. This is useful when dealing, for instance, with a source domain that has access to lots of resources (e.g. corpora, labeled datasets, external knowledge) and a target domain that lacks such resources. Here, we adopt a broad definition where "domain" refers to a "distribution over the feature space", as there is no clear consensus over what constitutes a domain (Plank, 2016). As a result, Domain Adaptation may be seen more generally as the practice of using available data from a source distribution to improve performance over some target dataset with a different distribution, usually within the same task (e.g. general-domain NLI $\longrightarrow$ medical NLI).

Historically, there have been several ways to categorize Domain Adaptation methods, and these have changed along with the notion of domain as well as the evolution of statistical learning (Jiang, 2008; Margolis, 2011; Weiss et al., 2016). One recent taxonomy however, that goes over traditional methods as well as more modern approaches (Ramponi and Plank, 2020), categorizes Domain Adaptation into: *model-centric* methods, which alter elements at the model-level such as the objective function, the model weights or the model architecture; *data-centric* methods, which focus on selecting relevant data (i.e. Data Selection), using a source model to generate labeled data (i.e. Pseudo-labeling) or using large unlabeled corpora and/or auxiliary tasks (i.e. Pre-training); and finally, *hybrid methods*, which leverage elements from both categories. In what follows, we will be focusing on the *data-centric* approach to Domain Adaptation, specifically, pre-training methods, as these are more relevant to the work we present in this manuscript.

Data Selection & Pseudo-labeling

Data-centric approaches to Domain Adaptation are able to leverage large datasets to build domain-specific systems. One such method is Data Selection, which aims to detect instances from the source dataset that are likely to be useful for training a model for the target domain. This has been mostly explored in Machine Translation for selecting a subset of parallel data that best matches the distribution of in-domain texts (Moore and Lewis, 2010; Yasuda et al., 2008), but has also been proposed for other tasks like sentiment analysis (Remus, 2012) and parsing (Plank and van Noord, 2011). In practice, a similarity metric is used to select examples from the source data as a pre-processing step before building a model for the target domain. Traditionally, this metric is either Language Model Perplexity, where a language model trained on in-domain data is used to score source instances; or Jensen-Shannon Similarity, which relies on the "distance"[10] between the source and target distributions.

Another data-centric approach is Pseudo-labeling, which has its roots in semi-supervised learning (Abney, 2007; Chapelle et al., 2009). Broadly speaking, Pseudo-labeling consists in leveraging source labeled examples to train an initial model, then leveraging this model along with target unlabeled examples to generate new labeled instances, enrich the original training set, and re-train the model again. This (hopefully) virtuous loop is repeated until model performance does not improve anymore. One of the simplest pseudo-labeling methods is *self-training* (Yarowsky, 1995), where a single model is used to generate labels for its own future training. However, this usually results in this model learning from its own mistakes which in turn can amplify errors, especially when the labeled and unlabeled data come from two different domains. An alternative to plain self-training is *co-training* (Blum and Mitchell, 1998), where this time, two models are used on conditionally independent feature representations of the original source data. If these representations are sufficient for training good models, then label generation and training are dissociated by having high-confidence labels from each model serve as additional examples for re-training the other model. However, finding feature representations that are both sufficient and independent is not always easy, and instead, a more practical alternative is to rely on the *tri-training* (Zhou and Li, 2005) approach. In this framework, three models are trained on different—usually bootstrapped—versions of the source data. Then, each model is re-trained on samples for which the other two models are highly confident.

Both Data Selection and Pseudo-labeling are still explored in recent work (Ma et al., 2019; Coleman et al., 2020; Arazo et al., 2020; Xu et al., 2020). However, as pre-trained language models took over modern NLP, pre-training has also become

[10]The Jensen-Shannon similarity is based on the Kullback–Leibler divergence. This measure is not technically a distance *per se* as it is not symmetric.

one of the most favoured methods for leveraging large amounts of data to adapt general models to specific domains.

Domain Adaptation Through Pre-training

We have touched upon Sequential Transfer Learning as an approach for leveraging available source data and learning general features that can be transferred across various downstream tasks (see Section 2.2.2). However, some instances of Sequential Transfer actually serve a domain adaptation purpose, learning features that are relevant for a specific target domain instead. In practice, these all involve an in-domain corpus or task during the pre-training phase and may therefore be seen as applying Domain Adaptation ideas—specifically, Data Selection[11]—in the context of Sequential Transfer Learning.

The simplest way to achieve domain adaptation through pre-training is to train models from scratch on in-domain data. There are several examples of such models, for instance, FlauBERT (Le et al., 2020) and AraBERT (Antoun et al., 2020) build language-specific BERT models for French and Arabic respectively by pre-training on monolingual corpora. Similarly, CamemBERT (Martin et al., 2020) and RobeCzech (Straka et al., 2021) train monolingual versions of RoBERTa (Liu et al., 2019) on French and Czech corpora respectively. Another example is PubMedBERT (Gu et al., 2020), which builds a specialized BERT model for the biomedical domain by training the model on a large biomedical corpus. However, training such models from scratch is not always practical and is originally the reason why Sequential Transfer was sought, learning and leveraging a set of general-purpose features. Consequently, instead of training models from scratch, some models use pre-trained models from the general domain to initialize in-domain pre-training in what can be essentially described as two consecutive model transfers: first, an initial pre-training that learns and transfers general features, then, a second in-domain pre-training that leverages these general features to learn and ultimately transfer domain-specific representations. Instances of such double transfer are numerous as well, namely: SciBERT (Beltagy et al., 2019) and FinBERT (Yang et al., 2020), which respectively train models for the scientific and financial domains using pre-trained weights from general-domain BERT; and RuBERT (Kuratov and Arkhipov, 2019), which trains a Russian BERT model by leveraging weights from the multilingual version of BERT.

Pre-training a model twice, once on a general-domain corpus then a second time on an in-domain corpus can be framed within the broader class of *multi-phase adaptive pre-training* methods (Ramponi and Plank, 2020). In fact, this particular instance of double pre-training is sometimes called Domain-Adaptive

[11] Usually, the data is selected manually by the NLP practitioner. However, this could also be achieved using automatic data selection methods.

Pre-Training (DAPT) and has been compared to a similar approach called Task-Adaptive Pre-Training (TAPT) where, instead of a second pre-training on large in-domain corpora, the language models are re-trained on smaller but more specific task datasets (Gururangan et al., 2020). While DAPT often performed better than TAPT, in practice, the authors show that both methods can be used jointly to improve performance even further in what is essentially a triple pre-training: a general-domain pre-training of the language model, then an in-domain pre-training using a large in-domain corpus and finally a task-specific pre-training using the target task corpus[12].

Similar to general-purpose Sequential Transfer, Model Transfer for Domain Adaptation has also been explored in both the unsupervised—specifically, the self-supervised—and supervised settings. For example, one instance that aims to leverage available labeled data in the source domain is AdaptaBERT (Han and Eisenstein, 2019), which pre-trains BERT on a corpus made of both modern English (source) and historical English texts (target), then fine-tunes the result on labeled POS tagging examples from the source domain. In this case, it is interesting to note that although the method includes a fine-tuning step, it may still be considered as a pre-training technique since the whole process is used in preparation for a model that will be transferred to the target domain (i.e. historical English)[13]. Another work that leverages labeled source data for domain adaptation is Supplementary Training on Intermediate Labeled data Tasks or STILTs (Phang et al., 2018), which was first introduced as a way to improve model transfer in general by leveraging source labeled data, but was then used for the purpose of the domain adaptation of multilingual models to specific languages (Phang et al., 2020). More specifically, the approach consists in using a pre-trained multilingual model (1^{st} source domain), fine-tuning on English labeled task data from one or multiple tasks (2^{nd} source domain), fine-tuning again on the English version of the target task, and finally transferring/evaluating the model on the target language version of this target task (target domain—e.g. Swahili, Russian, German). Similar to TAPT where a model is pre-trained in three steps on a general, in-domain and finally a task-specific corpus; this zero-shot cross-lingual application of STILTs applies the same principle of adapting a general-purpose model in progressively more specific settings but relying on source labeled data instead.

2.2.4 Summary

While in theory it is possible to train good models for any purpose given enough labeled examples, real life problems are often more chaotic and a number of differ-

[12]In practice, it is likely that only the task-specific pre-training will be necessary as there are many publicly available pre-trained models for various domains, including specialized domains.

[13]Recall that sequential transfer consists of two steps: a *pre-training* step and a *transfer* step.

ent issues may occur. Most often than not, the available data is scarce, and when enough data happens to be available, the future data cannot always be expected to come from exactly the same distribution. For all these reasons, many research efforts have tried to augment the pool of usable data by looking at related tasks and corpora. To this end, Transfer Learning aims to leverage such related resources to build models and learn features that can be re-used, enabling and improving target tasks. Among these methods, Sequential Transfer Learning (STL) is probably the most ubiquitous and is used today to pre-train large neural language models that are then fine-tuned to deal with a wide variety of NLP problems. Applied to specialized texts, this sequential transfer can also serve a domain adaptation purpose, learning domain-specific features instead of general ones. Nevertheless, text corpora are not the only resource that is available for building NLP systems. In fact, in the following section, we review a number of methods that use different kinds of resources, namely, knowledge bases and domain ontologies, which are leveraged to further enhance corpus-based word representations.

2.3 Leveraging External Knowledge

Today, word embeddings have become a key component for most modern NLP systems. These representations have proven to be effective on a wide set of applications, ranging from sentiment analysis (Socher et al., 2013) to entity recognition (Guo et al., 2014). However, some word embeddings still suffer from important limitations that are due, in part, to the fact that these vectors are usually trained following a tacit rule called the *distributional hypothesis* (Harris, 1954; Firth, 1957). This rule essentially states that in any given language, similar words will have a tendency to occur in similar contexts. As a result, it can be expected that analyzing a word's context may hint at its meaning. Accordingly, most embedding algorithms—and especially earlier methods like WORD2VEC and GLOVE—use some form of co-occurrence information to build their word representations[14]. These representations are then believed to encode some form of word meaning, however, in practice, word embeddings often encode information about word relatedness instead (Hill et al., 2015). For instance, words with similar meanings such as "like" and "love" usually appear in similar contexts (i.e. talking about appreciation), which leads to their representations being close in the embedding space. However, related but antonymous words such as "like" and "hate" would generally share practically identical contexts (e.g. "I liked this movie.", "I

[14]While more recent models indirectly rely on co-occurrences through language modeling, earlier methods either explicitly bias word vectors to be similar to their contexts' representations (e.g. WORD2VEC, FASTTEXT) or directly leverage co-occurrence statistics during training (e.g. GLOVE).

hated this movie"), thus leading to some antonyms having similar co-occurrence statistics to synonymous words and ultimately similar word representations. Consequently, word embeddings usually fail to encode precise meanings of words, and in some cases, word relatedness as well when such words appear infrequently in the training corpus. To address these issues, incorporating *external knowledge* has been explored as a way to enhance existing word representations with external information such as prior lexical (e.g. synonyms and antonyms) or domain knowledge (e.g. what drug cures which disease). Usually, this information is organized within structures like Knowledge Bases and Ontologies, which we briefly discuss in the following section.

2.3.1 Knowledge Bases and Ontologies

As often when dealing with definitions, there is no clear consensus on what constitutes a Knowledge Base. Generally speaking, however, a Knowledge Base (KB) may be defined as a structure containing and possibly organizing information about the world—or more reasonably, a specific domain. Technically, the term is believed to have been introduced to refer to one of two components which, along with an "inference engine", usually compose traditional expert systems (Hayes-Roth et al., 1983). In fact, knowledge bases have been used long before they were considered for enhancing corpus-based representations. In earlier days of Artificial Intelligence, when expert systems were relying on unstructured rule sets to solve complex problems, researchers soon came to the realization that these systems could scale up by benefiting from the sharing and re-use of a large amount of common knowledge (Pérez and Benjamins, 1999). As a result, many efforts have focused on building large knowledge bases that could support inference in what has come to be known as knowledge-based systems (KBS). In practice, these systems required the knowledge to be structured in a way that lends itself to logical inference (e.g. objects pointing to other objects as opposed to tables with numbers and strings), which has ultimately led to the development of domain ontologies.

Generally speaking, an Ontology is a conceptualization of a domain (Gruber, 1995) which often takes the form of a tree-like structure relating different concepts (e.g. "fever", "shivers") through a number of relations (e.g. "causes", "is caused by"). Since the early adoption of Ontologies as part of Expert Systems, these knowledge structures have been extensively used in other areas as well[15], namely: the Semantic Web (Berners-Lee et al., 2001), where domain ontologies were proposed to annotate Web pages in an attempt to enrich their contents with semantic information[16]; e-commerce, for matching users with best compatible goods and services

[15]For a non-exhaustive list of domain ontologies, you can refer to this Wikipedia page.

[16]For instance, a shopping list containing vegetables would not be a simple list of strings, but instead, a list of annotated elements with references to concepts such as "tomato" that would in

(Paolucci et al., 2003) or ensuring web security (Ekelhart et al., 2007); biomedicine, with ontologies like SNOMED RT (Spackman et al., 1997), NCIt (Golbeck et al., 2003), enabling several medical applications (Rector et al., 2019); and many others (Meenachi and Baba, 2012; Jain and Singh, 2013). However, building domain ontologies from scratch can be expensive and time consuming, usually requiring human experts from said domain, which has led to the development of the field of ontology learning (Hazman et al., 2011; Asim et al., 2018). Interestingly enough, we note that ontology learning methods sometimes rely on raw textual data in order to distill a structured form of knowledge (Omelayenko, 2001; Lenci et al., 2007), which is opposite to the current trend of corpus-based NLP models looking at such ontologies for information they were not able to learn from similar sources of text.

Today, Knowledge Bases and Ontologies are used in Natural Language Processing to enhance models with external knowledge. In practice, the reader may come across other kinds of knowledge structures such as Lexicons, Thesauri, Knowledge Graphs and Terminologies. While each structure has its own somewhat subtle and specific set of attributes that differentiate it from the rest, in the end, they all have in common that they contain prior knowledge which can be leveraged for improving corpus-based models and word representations in particular. In what follows, and for the sake of simplicity, we will often use the generic terms "knowledge base" and "external knowledge" to refer to any of the aforementioned structures, including domain ontologies.

2.3.2 Enhancing Traditional Embeddings

In this section, we will be going over the two main approaches for leveraging external knowledge to enhance traditional word representations (i.e. static embeddings), namely: Joint Specialization methods which add knowledge constraints during the training procedure, and Post-processing methods which inject this information into already pre-trained word vectors.

Joint Specialization: Injecting Constraints During Training

As previously discussed, external knowledge has been proposed to alleviate some of the shortcomings of traditional text-based embeddings which tend to encode ambiguous and incomplete information. Historically, earlier approaches to leveraging external knowledge adopted an intuitive strategy and simply altered the optimization process, injecting knowledge constraints into the objective function. Such approaches are known as Joint Specialization methods. For instance, the CLEAR method (Halawi et al., 2012) uses the WordNet knowledge base (Miller, 1998) to

turn have references to other concepts like "soup" or "agriculture" within the ontology.

extract pairs of related words (e.g. synonyms/antonyms or hypernyms/hyponyms) then biases word embedding learning to minimize the difference between these related words. Similarly, Yu and Dredze (2014) learn WORD2VEC representations from a news corpus while at the same time including linguistic constraints from WORDNET as well as a paraphrase database called PPDB (Ganitkevitch et al., 2013) so that semantically related words are encouraged to have similar representations. Other methods generalize this approach by including additional kinds of external knowledge, namely morphological and syntactic knowledge (Bian et al., 2014); or distinguishing relational (e.g. PARIS, `is_capital_of`, FRANCE) from categorical knowledge (e.g. {GB, GBR, UK} `is_alias_of` UNITED KINGDOM) (Xu et al., 2014).

However, only enforcing that related words and concepts be assigned similar representations is not always enough. In fact, we recall that words such as "love", "hate" and "like" are all related despite some being antonymous to others and some referring to a more intense feeling than others. This has led subsequent work to focus on word triples instead of word pairs with, for instance, (Liu et al., 2015) replacing the "A is related to B" constraints with *ordinal constraints* such as "A is more similar to B than it is to C". In practice, this is achieved by having the objective function include an additional term that optimizes the cosine similarity between A and B to be larger than the measure between A and C. Another work, which actually does not rely on an external knowledge base but rather extracts its constraints directly from text, is (Schwartz et al., 2015) where symmetric patterns (Davidov and Rappoport, 2006) are used to actively push away antonymous words. This approach was subsequently improved upon with Ono et al. (2015) using external knowledge from WORDNET and Roget's Thesaurus (Kipfer, 1993), and eventually a similar idea being reassessed by Nguyen et al. (2016).

Post-processing: Refining Embeddings With External Knowledge

Around the same time that methods were being developed for jointly specializing word representations using external knowledge, a new approach was focusing on refining existing representations in a post-processing fashion. Contrary to Joint Specialization, where a model has to be trained from scratch whenever additional knowledge constraints are desired, Post-processing methods further adapt existing models and therefore tend to be more lightweight. Moreover, these methods are generally agnostic to the kind of embedding technique that is used, as they only expect a set of word vectors as input and make no assumptions on how it was obtained.

The first influential post-processing method is probably *retrofitting*[17] (Faruqui

[17]There is another paper that appeared around the same time and proposed a similar approach called SENSERETROFIT (Jauhar et al., 2015). While this shares some ideas with the work by

et al., 2015), which casts the specialization problem as an optimization procedure where retro-fitted vectors are encouraged to be closer to related words in a knowledge graph while at the same time being prevented from diverging too much from the original representations. By expecting a knowledge graph, retrofitting homogenizes the expected format of external knowledge and only asks that it is expressed as a graph of nodes (e.g. words, concept) and edges (e.g. synonymy, hyponymy). More importantly, the retrofitting objective is convex, which makes it easier and faster to solve. Faruqui et al. (2015) conducted a thorough evaluation of retrofitting, testing various types pre-trained vectors (e.g. WORD2VEC and GLOVE), with multiple lexicons (e.g. WORDNET and PPDB) and on various tasks (e.g. synonym detection and sentiment analysis). In most cases, the post-processing method greatly improved over existing joint-specialization techniques. Naturally, this has led to several variations of retrofitting which attempted to further refine this approach.

One such variation is *counterfitting* (Mrkšić et al., 2016) which, seeing that retrofitting had only focused on word relatedness, and, in the same spirit as existing Joint Specialization methods that leverage synonym/antonym relations (Schwartz et al., 2015; Ono et al., 2015), proposed a different objective so that synonyms would attract each other, antonymous words would repel each other, while at the same time preserving existing structure (i.e. distances in the original embedding). Soon after, the counterfitting approach was further refined to leverage semantic intensity information which can be automatically extracted from texts (Kim et al., 2016a). In this framework, similar words with similar intensities are pulled together, and antonyms are used to optimize similar words with different intensities such that an antonym is always closer to a weaker word than it is to a stronger one.

One subsequent method was ATTRACT-REPEL (Mrkšić et al., 2017) which built on the counterfitting method as well as an earlier technique called PARAGRAM (Wieting et al., 2015) that had previously managed to refine word representations using paraphrase information. ATTRACT-REPEL introduced improvements to the overall optimization procedure and was not only applied for refining existing representations but also for building cross-lingual representations using multilingual lexical information from BABELNET (Navigli and Ponzetto, 2012; Ehrmann et al., 2014). In fact, by leveraging synonyms across multiple languages[18], this method is able to represent words from multiple languages in a joint embedding space that is then used to improve, for instance, tasks in low-resource languages (e.g. Hebrew and Croatian).

Building on the ATTRACT-REPEL method, an approach simply called *post-*

Faruqui et al. (2015), the latter is more general in the kind of external knowledge it considers.

[18] BABELNET compiles synonyms from different languages within structures called synsets.

specialization (Vulić et al., 2018) tackled the issue of words that do not appear in the external resource. In fact, all post-processing methods so far could only alter words for which there exists some information in the various thesauri and lexicons; for all other words, the representations would remain unchanged. In the post-specialization approach, the ATTRACT-REPEL method is first used in conjunction with external knowledge to learn retrofitted vectors for words appearing in the resource. Then, given the original and transformed vectors, a neural network is trained to reproduce the transformed representations from the original ones. Eventually, this transformation is applied to unseen words so that all words can be refined, which ultimately would lead to improved results.

In a similar spirit, a method called *explicit retrofitting* (Glavaš and Vulić, 2018) also aims to learn a global function for specializing the whole vocabulary of words, however, the difference lies in the way supervision is achieved: instead of relying on an initial specialization step to generate vector pairs and learn a global transformation, explicit retrofitting directly transforms the lexical constraints into regression examples by assigning to word pairs w_1, w_2, either a minimal similarity value for synonyms ($s_{\text{min}} = -1$), a maximal similarity for antonyms ($s_{\text{max}} = 1$), or—in order to preserve existing structure within the pre-trained vectors—the original similarity value ($s = cos(w_1, w_2) \in (s_{\text{min}}, s_{\text{max}})$) when no constraint exists for the word pair. Recently, more advanced architectures such as Triplet Networks were leveraged for a similar purpose (Wang et al., 2014; Schroff et al., 2015), improving performance even further (Shah et al., 2020).

2.3.3 Enhancing Pre-trained Language Models

While improving traditional representations with external knowledge has been extensively explored through both joint specialization and post-processing methods, modern NLP systems usually do not rely on static representations, but instead, tend to leverage large pre-trained language models that can produce more expressive contextual embeddings. Incidentally, accounts of successfully improving these language models using methods that merely adapt a pre-trained model in a post-processing fashion have been relatively sparse with, namely, Shi et al. (2019) retrofitting ELMO using paraphrase constraints to achieve varying degrees of improvement on downstream tasks. The scarcity of such approaches in the context of pre-trained language models may be due to their relative brittleness and susceptibility to catastrophic forgetting (Kirkpatrick et al., 2017; Arora et al., 2019). Nevertheless, there have been various efforts that achieved different degrees of knowledge injection through changes to the original model architectures and/or the training objectives.

In the following paragraphs, we briefly discuss the knowledge that seems to be already encoded within pre-trained language models. Then, we provide a non-

exhaustive list methods specializing these models with external models, grouping together techniques with similar approaches:

Knowledge Contained in Pre-trained LM Pre-trained language models have been shown to encode a great deal of linguistic knowledge (Goldberg, 2019; Lin et al., 2019), with lower layers usually encoding more syntactic information (Peters et al., 2018b; Manning et al., 2020) and the overall models encoding information about part-of-speech, syntactic chunks, as well as semantic roles, entity types, etc. (Tenney et al., 2019). Moreover, thanks to various probing efforts (Petroni et al., 2019; Roberts et al., 2020), pre-trained language models have been shown to often contain enough factual knowledge to be competitive with other methods relying directly on actual knowledge bases[19]. Nevertheless, similar to the traditional static representations, general-purpose language models are also able to benefit from some degree of specialization using external knowledge.

External Knowledge but Not a Knowledge Base One example of specializing pre-trained language models with external knowledge, which, incidentally, does not use an actual knowledge base, is SemBERT (Zhang et al., 2020). Instead, this model leverages an external Semantic Role Labeling model to generate semantic role representations which are ultimately concatenated with BERT's contextual vectors, presumably incorporating additional semantics into the language model. While not technically leveraging a knowledge base, this work is interesting in that it demonstrates that pre-trained models may also be used to enrich word representations with semantic information in a way that enhances these representations and boosts downstream performance.

Enhancing LMs by Altering the Input Text Another example of language models improved using external knowledge is K-BERT (Liu et al., 2020), which proposes various modules to inject knowledge base information into BERT models. Specifically, this method relies on a *knowledge layer* that converts any input sequence (e.g. Tim Cook is visiting Beijing) into a knowledge-rich sentence tree (e.g. Tim Cook {`CEO`, Apple} is visiting Beijing {`capital`, China; `is_a`, city}). This sentence tree is then fed to an *embedding layer* which generates context-independent token representations[20]. Then, a *seeing layer* produces a matrix that informs the model of which elements from the sentence tree are visible to each other. Finally, a different module called *mask-transformer encoder* uses this matrix to iteratively contextualize the sentence tree, similar to how *BERT* encoder

[19] A more detailed review of the different types of knowledge that are encoded within BERT-like models is available in (Rogers et al., 2020).

[20] Here, a token is an element of the tree rather than an actual token from the original sentence.

layers contextualize subword representations but only allow visible elements to attend to each other. Ultimately, the output vectors are used as features, improving performance on different downstream tasks when using knowledge bases from either the general or medical domain.

Enhancing LMs Using Knowledge Embeddings Focusing more on existing attention mechanisms, the KnowBERT approach (Peters et al., 2019) adapts BERT's multi-head attention to directly incorporate external knowledge into the model's hidden representations. Interestingly, this method uses the knowledge base implicitly by relying on knowledge base representations instead of KB triples. Moreover, the model uses an integrated entity linker that can be trained along with other model parameters during pre-training. In practice, for each knowledge base, KnowBERT adds a Knowledge Attention and Recontextualization component (KAR) to the bottom layers of BERT's architecture which essentially applies the following transformations: first, the incoming hidden representations are projected down to the same size as the pre-computed knowledge base embeddings; then, vectors within each mention span are pooled together to produce a single mention representation; all mention embeddings are contextualized through self-attention so that each mention can attend to all other mentions occurring in the input; in parallel, the integrated entity linker computes a weighted linear combination of all candidate entities' knowledge representations which results in a single knowledge vector for each mention from the input text; eventually, the contextualized mention representation is combined (through addition) with the aggregated knowledge vector, producing an updated set of mention representations that are used to re-contextualize the original hidden representations of all input tokens via self-attention[21]; finally, these "knowledge-enhanced" hidden representations are fed to subsequent Transformer layers—possibly going through the same process again if any other knowledge bases are used. While KnowBERT is probably among the most successful attempts at leveraging knowledge graph embeddings to inject external knowledge into transformer-based language models, other methods have adopted a similar approach with similar results. One such instance is ERNIE (Zhang et al., 2019), which proposes new encoder modules that can use entity embeddings trained on Wikidata[22] to contextualize BERT's representations. Instead of using the usual stack Transformer encoders, this architecture relies on stack of T-Encoders, which essentially act as a vanilla BERT model and produce contextual token representations; followed by a stack of K-Encoders, which iteratively apply the following transformations: first, the

[21] More specifically, the projected versions of the original hidden representations are contextualized before being projected up to their original dimension and fed forward.

[22] https://www.wikidata.org/

entity embeddings of mentions occurring in the input text are contextualized; then the token representations are contextualized as well, before both sources of information are combined using an *information fusion* module which essentially applies a dense layer on top of the concatenation of both vectors. Ultimately, the output representations are used to compute new token and entity representations which are eventually fed to the next K-ENCODER.

Enhancing LMs Using Knowledge Retrieval More recently, and in a slightly different line of work, a Retrieval-Augmented Language Representation Model pre-training paradigm (or REALM for short) was proposed (Guu et al., 2020), where a BERT-style "knowledge retriever" model is trained to retrieve relevant information from a large text corpus (e.g. WIKIPEDIA) in order to provide useful contexts that help a second TRANSFORMER model called "knowledge-augmented encoder" recover correct masked tokens during pre-training. In practice, this means that when the encoder model has to predict a masked token in a sentence like "*We paid twenty* [MASK] *at the Buckingham Palace gift shop.*", the knowledge retriever model would select information from the corpus such as "*Buckingham Palace is the London residence of the British monarchy.*" to disambiguate the context. As a result, the overall system is ultimately able to retrieve relevant information from a text corpus to support inference, which has been shown to bring notable improvements on tasks such as Open Question Answering. Subsequently, work by Agarwal et al. (2021) improved upon this REALM system, using a transformer-based text-to-text model to verbalize a large amount of Knowledge Base triples into a useful knowledge-rich corpus called KELM.

2.3.4 Combining Text and Knowledge Representations

In this section, we present *meta-embeddings*, an approach that consists in combining different sets of representations for achieving improved performance. First, we review how these meta-embeddings can be leveraged in a knowledge injection setting, combining text-based embeddings with knowledge representations. Then, we provide some context on how this approach is used in more general settings.

Meta-embeddings for Knowledge Injection

Some of the previously mentioned methods differ slightly from all other techniques in one particular aspect: they rely on pre-trained knowledge representations instead of using the knowledge base information directly. Such methods deserve a bit more focus as they allude to the general idea of *meta-embeddings*, where two (or more) different sets of representations are combined to produce a final set of improved embeddings. This combination may be achieved through different

means, and may serve different end goals. For instance, KnowBERT, which we mentioned previously, uses knowledge graph embeddings to contextualize BERT's internal representations. In this case, this process may be seen as using the self-attention mechanism to combine two sets of embeddings (i.e. BERT's hidden representations on one hand, and the knowledge embeddings on the other) for the purpose of enhancing the model with external knowledge. Another, perhaps more obvious, case of using meta-embeddings for leveraging external knowledge is (Goikoetxea et al., 2016), where the authors compare multiple ways to combine corpus-based word representations with knowledge embeddings obtained using a language model that is trained on randomly generated random walks over WordNet (Goikoetxea et al., 2015). In this work, the authors conclude that a simple concatenation of word and knowledge representations, with possibly an additional Principal Component Analysis (PCA) step, is sufficient to get results that outperform other specialization methods like retrofitting. This result is further confirmed by other experiments where, this time, PCA is shown to perform best (Jana and Goyal, 2018). In one instance, representations from three different modalities (text, knowledge base and images) are combined using several methods (average, concatenation, Singular Value Decomposition (SVD), etc.) and are shown to improve significantly over uni-modal representations on word similarity tasks (Thoma et al., 2017). Similarly, in the biomedical domain, representations built from medical knowledge bases were combined through concatenation with input word embeddings of neural systems and were shown to improve downstream performance (Wu et al., 2018; Jiang et al., 2019; Wang et al., 2019).

Meta-embeddings in the General Literature

In a context where dense co-occurrence based word representations had taken the NLP community by storm (Collobert and Weston, 2008; Mnih and Hinton, 2008; Huang et al., 2012; Mikolov et al., 2013; Pennington et al., 2014; Bojanowski et al., 2017), various efforts have considered ways to leverage different sets of word embeddings for building a single improved set of final representations. In fact, independently trained word representations seem to exhibit a great variance in the features that they learn (Chen et al., 2013) with, for instance, monolingual and bilingual models learning different types of semantics (Hill et al., 2014a,b). Naturally, model ensembles have been constructed for leveraging this variance, improving the overall downstream performance (Bansal et al., 2014). As a result, and based on previous work showing the benefit of using multiple sets of representations (Turian et al., 2010; Tsuboi, 2014; Luo et al., 2014; Yin and Schütze, 2015; Zhang et al., 2016), Yin and Schütze (2016) proposed a few simple methods for effectively constructing meta-embeddings from different sets of representations, namely: *concatenation*, where each vector set is normalized and possibly weighted

according to its expected contribution before all are concatenated; *SVD*, where the aforementioned concatenated vectors are projected down through singular value decomposition before being normalized; and *1toN*, a slightly more complex approach where randomly initialized meta-embeddings are trained to reproduce each singular embedding through a linear projection. Besides bringing consistent improvements over source vectors, these meta-embeddings have also been shown to alleviate out-of-vocabulary issues by leveraging representations with different levels of vocabulary coverage. Later, (Bollegala et al., 2018) proposed to learn meta-embeddings as local linear projections to leverage local information in the source embedding spaces. Here, local representations are learned as linear combinations of k-nearest neighbours in each source embeddings before being projected to a common global embedding space. Subsequent work explores simpler approaches involving linear combination. For instance, Coates and Bollegala (2018) demonstrate that meta-embeddings can be constructed by simply averaging representations from different source embeddings—with possible additional padding. This "frustratingly easy" method is then shown to perform on par with concatenation while keeping dimensionality under control. Kiela et al. (2018b) then generalize this idea and propose to use an attention mechanism (Bahdanau et al., 2015) in order to dynamically learn the linear combination weights. To handle source embeddings with different sizes, the authors propose to project each set of representations to a common space with the same dimensionality. Finally, more methods like *geometric meta-embeddings* (Jawanpuria et al., 2020) and *word prisms* (He et al., 2020b) propose further refinements by enforcing additional orthogonality constraints.

2.3.5 Summary

While it is natural to consider textual data when dealing with Natural Language Processing tasks, external knowledge in the form of ontologies, lexicons and more generally knowledge bases can also provide valuable information for further refining corpus-based systems. In this section, and with a particular focus on word representations, we have seen different approaches for leveraging such structured information. These methods vary according to the nature of the underlying word embeddings, with those applied to static representations generally constraining model training (Joint Specialization) or specializing existing representations (Post-processing), and those refining large neural language models generally relying on some form of modification to the model architecture and/or training objective. We have also discussed meta-embeddings—a general approach for combining multiple representations—highlighting its usefulness for incorporating external knowledge as long as it is pre-encoded in vectorial form. Incidentally, and while perhaps unconventional, we argue that knowledge injection methods can be framed within a

more general Transfer Learning context. For instance, Joint Specialization methods, by virtue of changing the training objective, may be seen as *model-centric* domain adaptation methods. Similarly, Post-processing techniques, given they fine-tune existing representations on auxiliary retrofitting-like tasks, may be seen as particular instances of domain adaptive—specifically, task adaptive—transfer. In the same spirit, *meta-embeddings* may be seen as combining elements from both *data-centric* domain adaptation (i.e. augmenting the model features) and sequential transfer (i.e. using pre-trained knowledge embeddings), making it, in some sense, a Transfer Learning technique as well.

2.4 Conclusion

In this chapter, we have reviewed a number of ways to specialize word representations by leveraging two different kinds of resources, namely: corpora and knowledge bases. By utilizing these resources together, it is expected to be able to construct NLP systems that go beyond the usual exploitation of texts, possibly including some extent of prior human knowledge as well. In the remainder of this manuscript, we will be largely relying on Transfer Learning tools—specifically, Sequential Transfer (e.g. for pre-training word representations) and *domain-adaptive* transfer (e.g. for training in-domain representations)—in order to build specialized models from scratch or refine existing general-purpose representations for specific target domains (e.g. the clinical and biomedical domains). In particular, we will be investigating how corpus size and domain similarity interact in this process, before proposing a method for building good in-domain representations while only leveraging a possibly small target task corpus. However, leveraging such corpora is only one side of the domain adaptation coin, as models trained using text corpora, both general and specialized, can only indirectly learn about actual world knowledge. Accordingly, we will also be exploring knowledge injection techniques and how these may be used in a domain adaptation setting to further refine word representations. Specifically, we will rely on meta-embeddings and will use them to refine both static and contextual types of representations. Moreover, and contrary to a number of existing knowledge injection techniques, we make it a point to adopt a simple approach which, arguably, enables the construction of strong baselines and simplifies any subsequent improvements.

Chapter 3

Embedding Evaluation via Downstream Performance

3.1 Introduction

Before delving into the more analytical topics of this thesis, we linger for a while on an important question of any experimental work: model evaluation. Specifically, we briefly review some of the different methods for evaluating word embeddings, providing our own perspective on the topic. Then, we go over the different ways that we have progressively adopted in this work in an effort to perform a fair and rigorous evaluation of our models.

3.2 Overview of Embedding Evaluation Methods

The NLP literature includes various methods for evaluating word embeddings (Bakarov, 2018) and these are generally categorized into two groups: *intrinsic* and *extrinsic* evaluations (Schnabel et al., 2015). Intrinsic evaluations consist in testing word representations directly for their semantic and syntactic contents, for instance, through word similarity (e.g. WS-353—Finkelstein et al. (2001)) or word analogy tasks (e.g. SAT—Turney et al. (2003)). These evaluations usually require some form of knowledge resource that serves as a gold standard and through which they are supposed to test the quality of word representations independently of any specific NLP task. Extrinsic evaluations, on the other hand, rely on target tasks such as sequence labeling (Xu et al., 2018), or text classification (Ravi and Ravi, 2015), and use the downstream performance on these tasks as a metric for evaluating the quality of the underlying embeddings. While extrinsic evaluations are generally criticized for being too task-specific and not providing insights that are general enough to reflect the inherent quality of tested word representations, in-

trinsic evaluations, on the other hand, are often criticized for providing information that is not strongly correlated with extrinsic performance (Tsvetkov et al., 2015; Faruqui et al., 2016).

Intrinsic or Extrinsic, Which is Better? Notwithstanding, we argue that intrinsic and extrinsic evaluations rest on a similar principle and that, in essence, both approaches are virtually the same: they both rely on a gold resource (e.g. thesauri, labeled classification datasets) and evaluate the ability of systems relying on different word representations to produce results that are consistent with said gold standard. In this context, the most notable difference between methods that traditionally fall into either the intrinsic or extrinsic categories probably resides in whether these systems rely on task supervision before the evaluation step. For instance, a bag-of-word sentence similarity system that averages word representations to produce sentence-level embeddings would be evaluated "intrinsically" if the underlying pre-trained embeddings have never been fine-tuned specifically for this similarity task, and "extrinsically" otherwise. In the end, a practical view of model evaluation would be that a good method is one that is best suited for the immediate need. In that sense, a good embedding evaluation anticipates the expected purpose of these representations and compares model performance on tasks that are likely to be performed with these word embeddings.

3.3 Our Approach to Embedding Evaluation

Throughout this thesis, we have grown progressively more sensitive to the question of embedding evaluation and have adopted, as a result, a number of practices in an attempt to perform a fair and careful evaluation of our representation systems. In the following paragraphs, we discuss the main techniques which we have used to perform these evaluations:

Multiple Random Restarts Computing a score distribution over multiple restarts and reporting the mean and standard deviation instead of a single score is unfortunately not a common standard in current NLP, especially in the context of large neural models, and despite different efforts advocating such a practice (Reimers and Gurevych, 2017; Lucic et al., 2018; Dodge et al., 2019). While reporting only a single performance can be harmful for the overall reproducibility of the results (i.e. giving wrong ideas about the expected performance of a model), this has also been shown to lead to false conclusions. For instance, Ma and Hovy (2016) proposed a sequence labeling system that was claimed to improve over a similar architecture by Lample et al. (2016). However, a more exhaustive exploration involving multiple random restarts of these models (Reimers and Gurevych,

2017) showed that the latter was actually better overall, with a score distribution that is both narrower (less variance) and higher (better scores on average). In order to avoid such biases in our own evaluations, we compute, whenever appropriate, a sample of the score distribution by running multiple (>10) random seeds for each model. Then, we report the mean and standard deviation for these distributions in an attempt to provide a fair and clear picture of the model behaviours. On a different yet related note, we also systematically train our models from scratch and in the exact same distribution despite similar models (e.g. general-domain BERT) being publicly available. This allows, in particular, to improve the fairness of our evaluations when running multiple seeds is not practical (e.g. when pre-training neural language models).

Model Ensembles Score distributions are useful and provide material for computing global performance statistics such as the sample mean and standard deviation, ultimately enabling a better interpretation of the results. However, computing these statistics is not the only way to aggregate the score information. In this thesis, we explore model ensembles as both a way to boost model performance and to summarize—to some extent—the score distribution. In fact, computing the average performance can be seen as a "downstream" approach to providing a single representative score for each model. In this context, computing ensembles may be seen as more of an "upstream" approach, combining models prior to the evaluation step. In practice, we rely on a simple voting strategy to combine the predictions of multiple models into a single one, keeping only the most frequently predicted label for token and sequence classification tasks, and averaging the model outputs for regression tasks. Moreover, we account for the variance of these ensembles as well and refrain from combining all seeds into a single ensemble. Instead, we build a set of ensembles for each model by excluding a single seed, computing an ensemble with the remaining seeds and repeating this process. Ultimately, we compute the mean and standard deviation of the ensemble distribution and use it for comparing our models.

Statistical Significance Tests These are probably the safest ways to compare models, provided that the score distributions respect the tests' underlying assumptions. There have been a growing interest for applying different statistical tests in NLP (Dror et al., 2018; Zhu et al., 2020), motivated by analyses showing the complex relationship between model performance and statistical significance, and advocating such formal tests as a general practice for validating metric gains in NLP (Berg-Kirkpatrick et al., 2012). In the context of this thesis, we would argue that computing average performance scores is a first step in the right direction. In fact, assuming the test distributions follow a normal distribution, these averages

could serve as the basis for *t-tests* (Student, 1908) which could provide information about the statistical significance of the difference between two models. However, in an effort to not make any particular assumptions that may bias our interpretation, we rely instead on Almost Stochastic Order tests (ASO) (Dror et al., 2019). These methods aim to determine whether a "stochastic order" exists between two models based on their respective sets of evaluation scores. In practice, given the set of performance scores of two models A and B, the method computes a test-specific value ϵ that indicates how far model A is from being significantly better than model B. Specifically, this distance ϵ is equal to 0 when model A $\succeq$ B, 1 when B $\succeq$ A, and 0.5 when no order can be determined.

3.4 Conclusion

Model evaluation is a complex matter, yet, its importance cannot be denied. In present-day NLP, however, evaluation is too often performed on the basis of single model results. These single scores can be harmful to the overall reproducibility of the results and may even lead to false conclusions. For these reasons, we adopt in this thesis a number of practices in an attempt to perform fair and—to some extent—rigorous system comparisons. These methods revolve mainly around using multiple seeds to produce score distributions that can be ultimately leveraged for computing global statistics (e.g. mean, std), model ensembles as well as statistical significance tests. Our approach, however, is not perfect and still has the potential to be refined further. One major way in which we could improve our methodology is to account for the fact that we sometimes make a large number of model comparisons. In fact, along this thesis, we progressively constitute a large set of evaluation tasks in an effort to provide a broader view of the different systems' strengths and weaknesses. However, these multiple comparisons compound the possible errors that may occur when comparing systems on a single task, thus increasing in a non-obvious way the chance of a misinterpretation. Future work may tackle this issue using Bonferonni correction (Dunn, 1961) as well as explore any other techniques for performing robust model evaluations on multiple tasks.

Chapter 4

Improving General Representations Using In-domain Corpora

4.1 Introduction

With word embeddings and neural architectures having become the go-to approach for solving complex NLP tasks, leveraging text corpora to construct these embeddings is today a common practice. Generally, texts from different sources are mixed together (e.g. news articles, encyclopedic entries, different genres of literature), constituting a somewhat hybrid domain that has come to be known as the "general domain" in order to compile sufficiently large corpora that allow the training of representations with good levels of performance. In fact, the downstream performance of word embeddings has been shown to positively correlate with the size of their training corpus, with successive leaps in performance being attributed to different methodological breakthroughs which have allowed the intake of larger and larger quantities of texts (Bengio et al., 2000; Collobert and Weston, 2008; Mikolov et al., 2013). Nevertheless, corpus size is not the only relevant parameter when pre-training word representations, especially when aiming for an application in a specialized context such as the medical domain. In fact, relying on corpora from similar distributions as the target domain can drive performance further, as these tend to exhibit similar patterns to those expected for the final task. As a result, a natural question arises about which of a larger corpus or a more domain-specific one is preferable when dealing with tasks in specialized domains. In the following section (§4.2), we examine this question in the context of the medical domain, comparing word representations built from corpora with varying sizes and degrees of specialization. However, in more practical situations, the specialized domains cannot always be reasonably expected to have access to large in-domain corpora and instead may only have the option to rely on the target task corpus

itself. In Section 4.3, we explore how this target task corpus can be fully leveraged using already pre-trained models, which we refine, as well as static representations that we build from scratch[1].

More generally, this work could be framed within the broader context of Transfer Learning as an investigation of how two specific parameters (i.e. corpus size and domain similarity) drive the quality of a domain adaptation achieved through pre-training. In fact, we recall (§2.2.2) that training word representations can be seen as an instance of sequential model transfer, where the model may be either a simple set of vectors (i.e. static representations) or an entire neural architecture (e.g. contextual language models). As such, it is expected that corpus size may have a positive impact on the quality of this transfer. However, when dealing with specialized domains, choosing an in-domain corpus becomes an option as well. In this case, the sequential transfer procedure begins to serve a domain adaptation purpose, learning domain-specific features instead of general-purpose representations. This leads to both corpus size and domain similarity being possible factors for driving the quality of word representations in specialized domains, which we explore in more details in the following sections.

4.2 Corpus Size vs. Domain Similarity

Corpus size and domain similarity are two key components of training domain-specific word representations. In practice, most systems default to using large in-domain corpora when such resources are available, as these allow to maximize performance along both the size and similarity axes. For instance, a survey in the medical domain involving over two hundred clinical NLP papers (Wu et al., 2020) shows that among the 63% of papers that used pre-trained representations, more than half used embeddings trained on large corpora such as MIMIC-III (Johnson et al., 2016), a corpus of clinical notes which we use extensively in this manuscript, and PubMed[2], a corpus involving a large number of biomedical article abstracts—which we use as well. However, despite the prevalence of large corpora in the medical domain, such resources may not always be available in other domains, leaving no other choice but to use general-purpose representations from the general domain. In this section, we investigate to what extent it may be problematic to use general-domain embeddings in a specialized context. More specifically, we train a number of clinical entity recognition systems and compare their downstream performance as a way to evaluate different embeddings trained on corpora with varying sizes and degrees of specialization.

[1]This work has led to a publication at ACL SRW 2019: (El Boukkouri et al., 2019).
[2]To learn more about PubMed visit: https://pubmed.ncbi.nlm.nih.gov/about/

4.2.1 Corpus Selection

We consider several corpora in an attempt to cover both the size and domain similarity spectra. First, we select two general-domain corpora:

GIGAWORD (Graff et al., 2003) These are news articles that have been compiled over multiple sources including, for instance, the New York Times. This is a large corpus (∼ 1 billion tokens) from the general domain for which we expect a negligible coverage of the medical field.

WIKIPEDIA These are encyclopedic articles from the 01/10/2017 data dump[3]. This corpus is large (∼ 1 billion tokens) and has possibly some limited coverage of the medical field. It is mainly, however, a general-domain corpus.

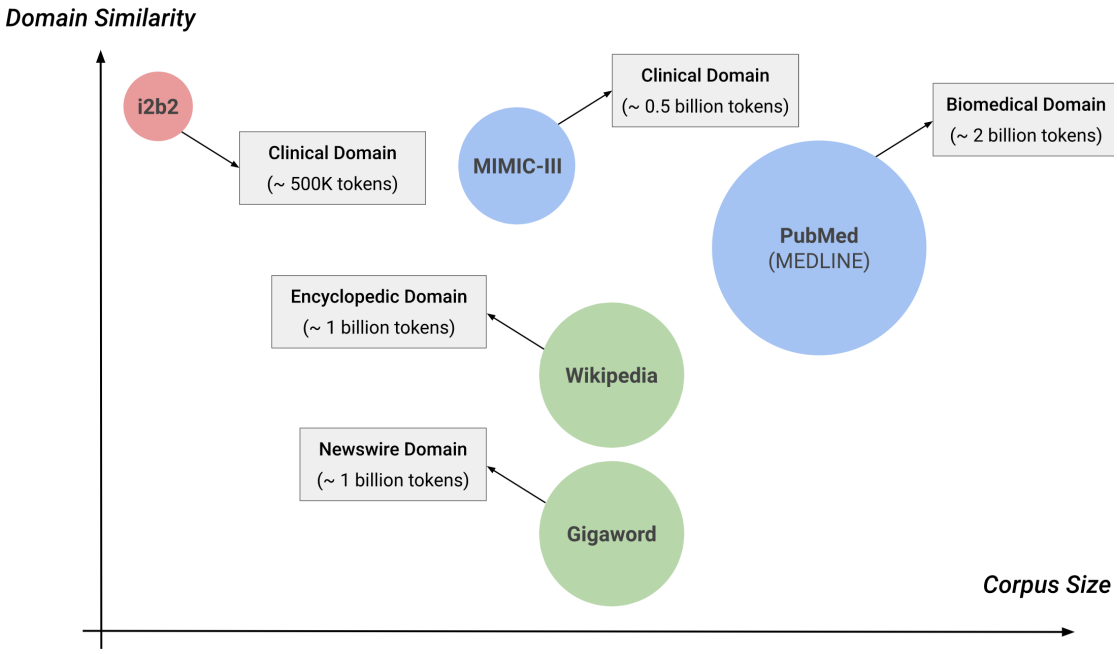

Figure 4.1: A visual summary of the different corpora considered for our experiments. Bubble sizes represent the corpus size and colors represent three categories of corpora: the target task corpus (in **red**), the general-domain corpora (in **green**) and the specialized corpora (in **blue**).

Then, we consider a small in-domain corpus:

I2B2 (Uzuner et al., 2011) These are clinical notes from the I2B2/VA 2010 clinical concept extraction challenge. We use texts from both the training

[3]Similar dumps can be downloaded at: https://dumps.wikimedia.org/enwiki/.

set as well as the provided unlabeled dataset, reaching $\sim$ 500K tokens. This is a specialized domain corpus—in fact it is our target domain—and we assume that a similar corpus is always obtainable using texts from the target task.

Finally, we consider two more in-domain corpora:

MIMIC-III (Johnson et al., 2016) This is a large collection of clinical notes from a database of Intensive Care Unit encounters[4]. The corpus is around 0.5 billion tokens and is from the clinical domain. Incidentally, the I2B2 texts are in part from MIMIC-III, making this corpus very close to the target domain.

PubMed (2018) This is a collection of biomedical article abstracts from PubMed-MEDLINE[5]. The corpus is quite large ($\sim$ 2 billion tokens) and is from a domain that is close yet somewhat different from our target domain (clinical).

These five corpora represent various positions in the corpus size vs. domain similarity space (see Figure 4.1). In fact, while **I2B2** is our most domain-specific corpus, it is also the smallest and may not provide enough contexts for training satisfactory representations. On the other hand, **Gigaword** and **Wikipedia** are significantly larger but may not have enough coverage of the medical field (i.e. vocabulary and contexts), which may induce biases in the resulting embeddings. Finally, **MIMIC-III** and **PubMed** are both medical corpora. However, the former is small and closer to the target domain while the latter is four times larger and from a slightly different domain (biomedical vs. clinical). It will be interesting to see whether given two corpora that are already quite large, size is worth as much as domain specificity.

4.2.2 Evaluation Through Clinical NER

Given our selection of corpora, we would like to investigate the impact of corpus size versus domain similarity on the quality of the resulting word representations. In order to assess this quality, we observe the downstream performance on an in-domain task for similar systems trained using the exact same hyper-parameters but leveraging different word embeddings. Our hypothesis is that by controlling all other parameters, any variation in the downstream performance will be due to variations in the inherent "quality" of the word representations and therefore, that "better" embeddings should result in improved performance and *vice versa*.

[4]The MIMIC-III corpus is available at: `https://mimic.physionet.org/`.
[5]The PubMed (MEDLINE) corpus can be downloaded at *this url*.

I2B2/VA 2010 Clinical Concept Extraction

All our systems are evaluated on the Clinical Concept Detection task from the 2010 I2B2/VA challenge (Uzuner et al., 2011). The task data consists of discharge summaries and progress reports from three different institutions: Partners Healthcare, Beth Israel Deaconess Medical Center, and the University of Pittsburgh Medical Center. These documents are labeled and split into 394 training files and 477 test files for a total of 30,946 + 45,404 ≈ 76,000 labeled sequences[6]. Given these datasets, the goal is to extract three types of medical entities: PROBLEM (e.g. "headache"), TREATMENT (e.g. "oxycodone") and TEST (e.g. "MRI").

Table 4.2 provides some examples of entity mentions and Table 4.1 shows the distribution of each entity type in the training and test datasets.

Entity type	Train set	Test set
PROBLEM	11,967	18,550
TREATMENT	8,497	13,560
TEST	7,365	12,899
Overall total	27,829	45,009

Table 4.1: Distribution of medical entity types in the I2B2 task.

3. **Echocardiogram** on **DATE[Nov 6 2007] , showed **ejection fraction** of 55% , **mild mitral insufficiency** , and **1+ tricuspid insufficiency** with **mild pulmonary hypertension** .
DERMOPLAST TOPICAL TP Q12H PRN **Pain** **DOCUSATE SODIUM** 100 MG PO BID PRN **Constipation** **IBUPROFEN** 400-600 MG PO Q6H PRN **Pain**
The patient had **headache** that was relieved only with **oxycodone** . **A CT scan of the head** showed **microvascular ischemic changes** . A **followup MRI** which also showed **similar changes** . This was most likely due to **her multiple myeloma** with **hyperviscosity** .

Table 4.2: Three labeled sequences from I2B2 showcasing different types of entity mentions (PROBLEM, TREATMENT, and TEST). Originally from (Roberts, 2016).

Word Embeddings

For each corpus, we train three sets of word representations using the three standard static embedding methods: WORD2VEC (Mikolov et al., 2013), GLOVE (Pennington et al., 2014) and FASTTEXT (Bojanowski et al., 2017). We consider different embedding methods in an effort to control for specific biases in these algorithms

[6]Due to limitations introduced by the Institutional Review Board (IRB), only part of the original data is now available for research purposes at: https://www.i2b2.org/NLP/DataSets/. Our work, however, uses the full original dataset.

that may affect our evaluation. For each method, the implementation and hyper-parameters which we use are mostly standard:

WORD2VEC We use the official implementation[7] to train 256-dimensional vectors using the CBOW variant, with a window size of 5, ignoring tokens that appear less than 5 times, and going over the entire corpus 10 times.

GLOVE We train 256-dimensional vectors using the official implementation[8], ignoring tokens that appear less than 5 times and going over the entire corpus 10 times. However this time we use a window size of 15.

FASTTEXT Here again, we use the official implementation[9] to train 256-dimensional vectors. This time we use the Skip-Gram method with negative sampling, using a context window of 5 with 5 negative samples. We also ignore tokens that appear less than 5 times, use n-grams with $n = 3$ up to $n = 6$ and go over the corpus 10 times.

Other than the aforementioned parameters, all other hyper-parameters (e.g. learning rate) are set to their default values.

Experimental Setup

After pre-training all the word embeddings, we use each set of representations to initialize a clinical entity recognition system which is used as a proxy for evaluating and comparing the contribution of each embedding method, and more importantly, each underlying corpus. We use a rather standard entity recognition system that consists in a Bi-LSTM CRF architecture (Huang et al., 2015), since this has been shown to perform well on various sequence labeling tasks (Plank et al., 2016; Lample et al., 2016; Habibi et al., 2017). Specifically, we use:

- a set of word embeddings, which we train along with the rest of the network;
- three bidirectional LSTM layers with 256 hidden units. Note that the overall number of units is 512 as each uni-directional LSTM has a size of 256;
- a recurrent dropout rate of 50%. This means that for each forward pass, half of the activations between each pair of Bi-LSTMs are randomly masked to avoid over-fitting. This only occurs during training, however, as all activations are used during the evaluation step;

[7] WORD2VEC implementation: https://code.google.com/archive/p/word2vec/
[8] GLOVE implementation: https://nlp.stanford.edu/projects/glove/
[9] FASTTEXT implementation: https://github.com/facebookresearch/fastText

- a CRF (Lafferty et al., 2001), which uses the output of the last Bi-LSTM layer as input features to predict token labels in BIO format.

For convenience, we rely on the AllenNLP framework (Gardner et al., 2018) to define and train our entity recognition systems. All models are trained using an early-stopping procedure that consists in monitoring performance on a validation set built from 20% of the original training data and terminating training when this performance no longer increases. More specifically, we use a patience value of 3 (i.e. we stop training after three unsuccessful iterations) and monitor the exact-span F1 measure since it is the official metric for the i2b2 clinical concept extraction task. All models are trained multiple times using a total of 10 random seeds in an effort to perform a fair evaluation and in accordance with the approach previously discussed in Chapter 3.3.

4.2.3 Results & Discussion

Corpus	Exact Match F1
word2vec	
i2b2	82.06 ± 0.32
Gigaword	82.54 ± 0.41
Wikipedia	83.30 ± 0.25
PubMed	84.06 ± 0.14
MIMIC	**84.29 ± 0.30**
GloVe	
i2b2	80.21 ± 0.37
Gigaword	81.38 ± 0.33
Wikipedia	81.82 ± 0.52
PubMed	**82.38 ± 0.57**
MIMIC	**82.38 ± 0.43**
fastText	
i2b2	81.98 ± 0.41
Gigaword	81.77 ± 0.36
Wikipedia	82.32 ± 0.37
PubMed	82.23 ± 0.49
MIMIC	**82.73 ± 0.39**

Corpus	Precision	Recall	F1
	WORD2VEC		
I2B2	82.03 ± 0.69	82.08 ± 0.42	82.06 ± 0.32
GIGAWORD	82.26 ± 0.72	82.82 ± 0.64	82.54 ± 0.41
WIKIPEDIA	83.42 ± 0.78	83.20 ± 0.91	83.30 ± 0.25
PUBMED	**83.87 ± 0.60**	84.26 ± 0.43	84.06 ± 0.14
MIMIC	**83.83 ± 0.77**	**84.77 ± 0.52**	**84.29 ± 0.30**
	GLOVE		
I2B2	80.29 ± 1.19	80.15 ± 0.71	80.21 ± 0.37
GIGAWORD	81.18 ± 0.60	81.58 ± 0.77	81.38 ± 0.33
WIKIPEDIA	81.35 ± 1.01	82.31 ± 0.49	81.82 ± 0.52
PUBMED	82.09 ± 1.17	**82.69 ± 0.45**	**82.38 ± 0.57**
MIMIC	**82.35 ± 1.03**	82.43 ± 0.64	**82.38 ± 0.43**
	FASTTEXT		
I2B2	82.85 ± 1.26	81.14 ± 0.81	81.98 ± 0.41
GIGAWORD	81.48 ± 1.17	82.07 ± 0.84	81.77 ± 0.36
WIKIPEDIA	82.44 ± 1.03	82.22 ± 0.83	82.32 ± 0.37
PUBMED	81.85 ± 1.48	**82.66 ± 1.03**	82.23 ± 0.49
MIMIC	**82.94 ± 1.07**	82.54 ± 0.78	**82.73 ± 0.39**

Table 4.3: Average clinical entity recognition performance of systems leveraging embeddings trained on different corpora. Best precision, recall and F1 measure is shown in **bold** and second best is underlined.

Table 4.3 reports the performance of trained clinical entity recognition systems as mean ± std computed over 10 random seeds. As expected, there is some variability in the results depending on which embedding method is used. However, there also seems to be a number of global patterns that emerge, namely:

General-domain < In-domain Overall, we can see that the performance of general-domain corpora is overall lower than that of in-domain (i.e. clinical and biomedical) corpora. For instance, we see that PUBMED and MIMIC respectively score an average of 84.06 and 84.29 strict F1 when using WORD2VEC compared to 82.54 and 83.30 on average for GIGAWORD and WIKIPEDIA. Such behavior is expected and confirms existing results in the literature (Roberts, 2016).

Small In-domain < Large General-domain We also see that the target task corpus, which is the most domain-specific but also many orders of magnitude smaller than the other corpora, usually leads to the worst performance (see I2B2 vs. WORD2VEC and GLOVE). This is interesting as it shows that domain similarity alone is not enough for building good in-domain representations. We also note that while I2B2 performs worst in most cases, it is nevertheless on par with large general-domain corpora when using FASTTEXT. This could hint at methods based on subwords being able to better leverage smaller-sized corpora, however, this can also be due to sub-optimal hyperparameter choices that could have prevented other corpora from being leveraged properly (see FASTTEXT vs. WORD2VEC).

Differences among General-domain Models Looking a bit closer, we can notice that WIKIPEDIA usually leads to better results than GIGAWORD. While both corpora are from the general domain, given the encyclopedic nature of WIKIPEDIA it can be expected that this corpus may have a better coverage of the medical field than GIGAWORD. In fact, a comparison of tokens occurring in I2B2 but which are out-of-vocabulary in the general corpora shows 5.82% OOV tokens when using WIKIPEDIA embeddings versus 14.42% when using GIGAWORD. These results serve as a reminder that so-called "general domain" corpora often include a number of different sub-domains, and hint at general corpora with different domain compositions being potentially more relevant in certain situations than others.

Very Large Biomedical < Large Clinical Another interesting result is that systems based on MIMIC often outperform those using PUBMED embeddings despite both corpora being from the medical domain and the latter being four times as large. This shows that while corpus size matters (e.g. I2B2 vs. MIMIC), it may be less important than domain similarity when sufficient size has already been achieved. In fact, we recall that MIMIC is from the same clinical domain

as I2B2, which makes it more similar than the biomedical articles from PUBMED. Naturally, this raises the question of what constitutes a sufficiently large corpus, which is not at all straightforward and still remains to be answered.

4.3 Leveraging the Target Task Corpus

We have seen previously that both corpus size and domain similarity matter when choosing a corpus for building domain-specific word representations. However, in practice, most domains do not have access to large in-domain corpora such as MIMIC-III or PUBMED for building word embeddings with satisfactory levels of performance. Moreover, we have seen that I2B2, a small corpus from the target task, was not sufficient for training representations that are competitive with such large specialized corpora, nor was it able to improve over representations produced from large corpora from the general domain. In this section, we tackle this issue and attempt to compensate for the unavailability of sufficiently large in-domain corpora by leveraging a (possibly small) target task corpus along with publicly available resources such as general-domain pre-trained language models. More specifically, and while remaining in the context of the I2B2 clinical concept extraction task, we explore how a combination of fine-tuning pre-trained representations and training new ones from scratch can be employed to make the most of the target task corpus, hopefully reaching results that are on par with word embeddings trained on large specialized corpora (i.e. MIMIC and PUBMED).

4.3.1 Embedding Strategies

The ways that a small corpus from the target task can be leveraged to build word representations are not numerous. In the present section, we explore a couple of different approaches:

1. we train representations from scratch directly on said corpus;
2. we use this corpus to further pre-train a set of existing representations.

In both cases, there is a choice to be made: either using static representations or contextual language models. Since the target task corpus is assumed to be rather small, the representations we train from scratch are static embeddings, as larger contextual models would probably require bigger corpora in order to be trained successfully. On the other hand, we choose to further pre-train a neural language model as these are supposedly able to encode more information than static representations, making them a better starting point for a domain adaptive pre-training. These are the model configurations that we consider:

Static Embeddings

We re-use the static representations from our previous experiments. These are embeddings trained using WORD2VEC, GLOVE or FASTTEXT on five different corpora: two from the general-domain (i.e. GIGAWORD and WIKIPEDIA), two from the medical-domain (i.e. MIMIC and PUBMED) as well as the target task corpus (i.e. I2B2). In practice, specialized domains may not have access to large in-domain corpora. However, we include these in our experiments as baselines for best-case scenario performance. Similarly, the static representations trained on general corpora are included as baselines, and are meant to be compared with general-purpose pre-trained language models.

Pre-trained Language Models

We choose ELMO (Peters et al., 2018a) as our pre-trained language model[10]. Specifically, we consider different off-the-shelf versions of this model[11]:

ELMO(small) This is a general-domain model that has been trained on the 1 Billion Word Benchmark corpus (Chelba et al., 2013). It is the smallest version of ELMO and produces 256-dimensional embeddings.

ELMO(original) This is a general-domain model trained on a mixture of WIKIPEDIA and newswire data. It produces 1024-dimensional embeddings.

Each model is further pre-trained[12] on language modeling for a number of epochs on the I2B2 corpus in an attempt to adapt their representations to the target domain. At each epoch, we monitor the perplexity achieved on a validation corpus (see Figure 4.2). Ultimately, the best model is retained:

ELMO(small)$_{finetuned}$ the result of fine-tuning ELMO(small) for 10 epochs. Here, we stop at 10 epochs as the model starts to overfit if trained any longer.

ELMO(original)$_{finetuned}$ the result of fine-tuning ELMO(original) for 5 epochs. Here, we fine-tune the model for fewer epochs compared to the ELMO(small) as this larger model is more prone to overfit (see Figure 4.2 at epoch 5).

Additionally, we consider a medical version of ELMO that was pre-trained on the biomedical corpus PUBMED to serve as a baseline: **ELMO(PUBMED)**.

[10]Here, we chose ELMO instead of BERT (Devlin et al., 2019) since, at the time of these experiments, BERT was still new and somewhat misunderstood. Moreover, ELMO was easier to experiment with, namely due to its word-level nature.

[11]Models and model descriptions are available at: https://allennlp.org/elmo

[12]Using the official implementation at: https://github.com/allenai/bilm-tf

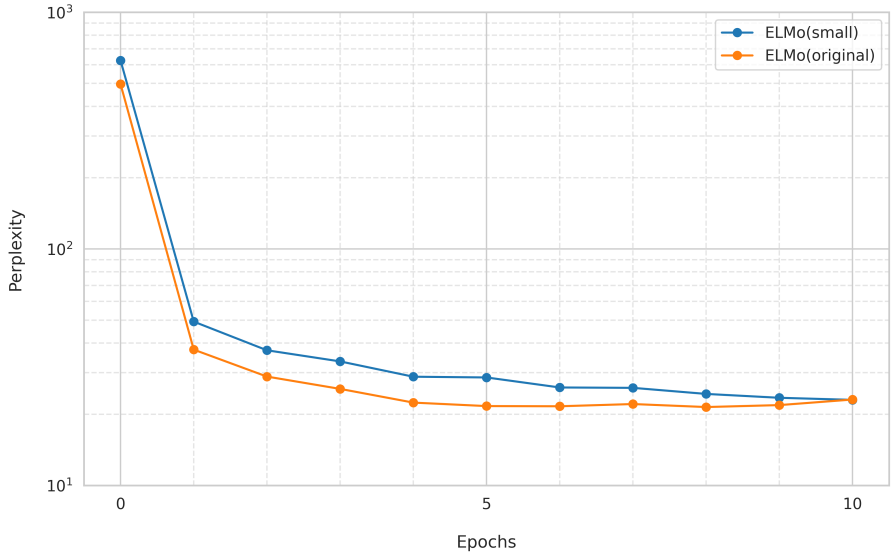

Figure 4.2: Language modeling perplexity during the fine-tuning of ELMo on I2B2.

Embedding Combinations

Word representations coming from different corpora and/or originating from different methods have already been combined in order to achieve improved downstream performance (see Section 2.3.4). To make the most of the target task corpus, we explore different combinations involving on one hand, the static representations built on the target task corpus itself; and on the other hand, general representations or versions fine-tuned on the task corpus. In practice, there are many ways to combine two representations. However, for the sake of simplicity, we consider two simple, yet effective methods:

Concatenation This is a simple concatenation of embeddings coming from two different models. This is denoted $\mathbf{X} \oplus \mathbf{Y}$ (e.g. I2B2 $\oplus$ WIKIPEDIA).

Mixture In the specific situations where ELMo is combined with static vectors, we can leverage ELMo's architecture to directly incorporate the static representations through linear combination. We denote this combination $\mathbf{X} +\!\!+ \mathbf{Y}$ (e.g. I2B2 $+\!\!+$ ELMo(small)).

The mixture method generalizes the way ELMo representations are combined. In fact, we recall (see Appendix A.2.1) that given a word w, if we denote the three internal representations produced by ELMo (i.e. the CHARACTERCNN, 1^{st} Bi-

LSTM and 2^{nd} Bi-LSTM representations) by h_1, h_2, h_3, we recall that the model computes the final embedding as:

$$\text{ELMo}(w) = \gamma \cdot (\alpha_1 \cdot h_1 + \alpha_2 \cdot h_2 + \alpha_3 \cdot h_3) \tag{4.1}$$

where $\gamma, \alpha_1, \alpha_2$ and α_3 are tunable task-specific coefficients[13]. Given h_{static}, the static representation of the word w, we compute a "mixture" representation as:

$$\text{ELMo}_{\text{mix}}(w) = \gamma \cdot (\alpha_1 \cdot h_1 + \alpha_2 \cdot h_2 + \alpha_3 \cdot h_3 + \beta \cdot h_{\text{static}}) \tag{4.2}$$

where β is a new tunable parameter[14].

4.3.2 Experimental Setup

We keep the same experimental setup as in Section 4.2.2, namely: we evaluate BiLSTM+CRF systems on the I2B2/VA clinical concept detection task and compare these models as a proxy to evaluate the quality of our embedding strategies. Strategies based on ELMO do not fine-tune the model but instead use it in a feature extraction setting (i.e. as a feature generator), making it essentially behave like a set of static embeddings[15]. As with previous experiments, we run a number of random restarts and report the average and standard deviation of the score distribution as our final results (see Chapter 3 on model evaluation). We also re-use the same training and evaluation parameters as Section 4.2.2.

4.3.3 Results & Discussion

We report the performance of clinical entity recognition systems according to their underlying embedding strategy. Given static representations trained on the target task corpus (i.e. I2B2), we report three different strategies:

- single sets of embeddings ($\mathbf{X}$);
- concatenations of I2B2 vectors and other representations ($\mathbf{\text{I2B2}} \oplus \mathbf{X}$);
- mixtures (i.e. internal linear combinations) of I2B2 and ELMO ($\mathbf{\text{I2B2}} + \mathbf{X}$).

Given the large number of experiments, we organize our results in multiple sections (see Table 4.4). First, there are three main sections which refer to results obtained with static representations trained using either WORD2VEC, GLOVE or

[13] In practice, the coefficients go through a softmax before being used in the linear combination.

[14] In specific cases where the ELMO model produces 1024-dimensional embeddings, we duplicate the 256-dimensional WORD2VEC embeddings so that the dimensions match before mixing.

[15] It is important to note that while ELMO's hidden representations are fixed during training, the linear combination weights, however, are trainable.

Embedding Strategy	**X**	**I2B2 ⊕ X**	**I2B2 + X**
		WORD2VEC	
I2B2	82.06 ± 0.32	-	-
WIKIPEDIA	83.30 ± 0.25	83.35 ± 0.62	-
GIGAWORD	82.54 ± 0.41	83.10 ± 0.37	-
ELMO(small)	80.79 ± 0.95	84.18 ± 0.26	84.94 ± 0.94
ELMO(original)	84.28 ± 0.66	85.25 ± 0.21	85.64 ± 0.33
ELMO(small)$_{finetuned}$	83.86 ± 0.87	84.81 ± 0.40	85.93 ± 1.01
ELMO(original)$_{finetuned}$	85.90 ± 0.50	86.18 ± 0.48	**86.23 ± 0.58**
		GLOVE	
I2B2	80.21 ± 0.37	-	-
WIKIPEDIA	81.82 ± 0.52	81.29 ± 0.42	-
GIGAWORD	81.38 ± 0.33	81.47 ± 0.18	-
ELMO(small)	80.79 ± 0.95	83.04 ± 1.03	84.30 ± 0.72
ELMO(original)	84.28 ± 0.66	85.00 ± 0.32	85.12 ± 0.26
ELMO(small)$_{finetuned}$	83.86 ± 0.87	84.42 ± 0.75	85.19 ± 0.75
ELMO(original)$_{finetuned}$	85.90 ± 0.50	86.05 ± 0.16	**86.46 ± 0.36**
		FASTTEXT	
I2B2	81.98 ± 0.41	-	-
WIKIPEDIA	82.32 ± 0.37	81.84 ± 1.48	-
GIGAWORD	81.77 ± 0.36	82.40 ± 0.32	-
ELMO(small)	80.79 ± 0.95	84.44 ± 0.42	85.47 ± 0.61
ELMO(original)	84.28 ± 0.66	85.57 ± 0.46	85.77 ± 0.47
ELMO(small)$_{finetuned}$	83.86 ± 0.87	85.18 ± 0.67	86.27 ± 0.35
ELMO(original)$_{finetuned}$	85.90 ± 0.50	86.49 ± 0.28	**86.82 ± 0.29**
		Baselines	
WORD2VEC(PUBMED)		84.06 ± 0.14	
WORD2VEC(MIMIC)		84.29 ± 0.30	
ELMO(PUBMED)		86.29 ± 0.61	
ELMO(PUBMED) + MIMIC		87.17 ± 0.54	
ELMO(Clinical) (Zhu et al., 2018)		86.84 ± 0.16	
ELMO(MIMIC) (Si et al., 2019)		**87.80**	

Table 4.4: Performance of various strategies involving a general-domain resource and a small in-domain corpus (I2B2). We report the exact span F1 scores as mean ± std with best values in **bold** and second best underlined.

fastText. In each of these sections, there are three subsections which respectively show the combination results that involve static embeddings trained on the target task corpus (i.e. I2B2) combined with either general static representations (i.e. Wikipedia or Gigaword), general pre-trained language models (i.e. ELMo) or fine-tuned versions of these language models (i.e. ELMo$_{finetuned}$). Finally, the last main section reports baseline results using large in-domain corpora that come from either our own experiments or from existing literature. All in all, our results lead to a number of insights which we discuss in the following:

Target Task Corpus < Large Corpora We confirm that training static representations on the target task corpus is usually a sub-optimal, with most models, including static embeddings trained on large general-domain corpora, leading to better results. This further showcases the difficulty in low-resource settings and reaffirms the need for better alternatives.

ELMo(small) < All Other Models It is interesting to note that ELMo(small) is almost systematically below all other models, including static I2B2, Gigaword and Wikipedia vectors. One possible explanation for this is that neural language models such as ELMo may require large enough architectures in order to perform optimally in a way that is superior to standard static representations. Another one is that models with smaller architectures such as ELMo(small) may not be suited to be used in a feature extraction mode (i.e. not entirely fine-tuned on the downstream task) when applied on a different domain than the one they were originally trained for.

Concatenation Improves over Base Models We can see that a simple concatenation systematically leads to improved performance across all base models (i.e. I2B2 $\oplus$ **X** $\geq$ **X**) with very few minor exceptions. These improvements are often small when combining I2B2 with general static representations like Gigaword and Wikipedia. However, it is interesting to note that the models which seem to benefit the most from this strategy are the pre-trained language models ELMo. In fact, we can see that both ELMo(original) and ELMo(small) are affected positively by a concatenation with I2B2, with ELMo(small), in particular, improving by several F1 points. However, this phenomenon could be explained in part by the fact that ELMo(small) is probably sub-optimal for our specific domain, as results obtained with this model alone are often lower than those of general-domain static representations. Nevertheless, the concatenation of ELMo(small) with I2B2 is fruitful enough to enable the language model to largely outperform general static representations, making this strategy an interesting approach to leveraging small-sized pre-trained models in a specialized and low resource context. More generally,

concatenating static representations trained from scratch on a (rather) small corpus from the target domain with readily available models from the general domain seems to be an effective method to improving downstream results when no large in-domain corpora are available.

Better Combinations with FASTTEXT? Another interesting result is that the I2B2 representations built with FASTTEXT, which are originally on par with those of WORD2VEC (i.e. 81.98 F1 vs. 82.06 F1 respectively), somehow manage to result in slightly better combinations with ELMO models (see WORD2VEC vs. FASTTEXT for I2B2 and ELMO combinations). While this may be true on average, this difference may not be significant in practice given the observed variance of the results. For instance, we observe that ELMO(small)$_{finetuned}$ achieves 85.93± 1.01 with WORD2VEC vectors and 86.27± 0.35 with FASTTEXT. This shows the importance of running repeated experiments with different seeds in order to compute both measures of location (i.e. mean) and dispersion (e.g. std).

Beneficial Pre-training on the Target Task Corpus If we look at the fine-tuned versions of ELMO which were further pre-trained on the target task corpus, we can see that this fine-tuning alone is able to result in systematic increases in performance. This shows that such large pre-trained models can benefit from further in-domain pre-training and demonstrates that a possibly small target-task corpus can be enough for achieving successful domain adaptation of pre-trained language models. These results echo subsequent findings in more recent work which shows, in the context of BERT-like models, that additional task-specific pre-training is usually beneficial (see TAPT in Section 2.2.3).

Best Strategy: Fine-tuned ELMO and In-Domain Static Vectors More importantly, it seems that fine-tuned versions of ELMO can also be combined through concatenation with static representations built from the target task corpus in what seems to be an even superior embedding strategy. In fact, this seems to be the best method so far, second only to replacing concatenation with an internal mixture with ELMO's hidden states. The mixture shows systematic improvements over the concatenation as well, although these gains are relatively small. However, it is important to note that the mixture combinations (i.e. I2B2 + $\mathbf{X}$) preserve the original embedding size while concatenation results in larger representations. With that in mind, it is safe to say that the best embedding strategy consists in training representations from scratch on the target task corpus (i.e. I2B2), fine-tuning existing pre-trained language models on the same task corpus (i.e. ELMO$_{finetuned}$) then incorporating the static vectors through linear combination into the freshly fine-tuned language model. This way, we are able to achieve a 4–6 points increase

in exact span F1 compared to training static embeddings on the target-task corpus alone, as well as similar improvements compared to using pre-trained language models from the general domain, with an additional 5–6 F1 for ELMo(small) and 2–3 F1 points for the larger ELMo(original).

Comparison with In-domain Baselines It is interesting to see that large general-domain versions of ELMo perform on par with static representations built on large in-domain corpora (see ELMo(original) vs. word2vec(PubMed-/MIMIC)). This could mean that any efforts intended for training in-domain representations may be better leveraged if focused on adapting large pre-trained models instead of training new static vectors from scratch. However, we expect this to be highly dependent on the kind of task at hand, with intrinsic (e.g. word relatedness) and extrinsic types (e.g. sequence labeling) having usually different embedding requirements. Nevertheless, this means in practice that our embedding strategies are able to perform better than static representations trained on large in-domain corpora without having to rely on such resources. The overall best performance we get using our methods is 86.82 ± 0.29 F1 with fastText vectors. If we compare this performance to large language models that have been pre-trained on large in-domain corpora, we can see that we are actually able to perform better than ELMo(PubMed) and perform on par with another version trained on a mix of MIMIC-III clinical reports and medical Wikipedia articles (i.e. ELMo(clinical) (Zhu et al., 2018)). Nevertheless, we are not able to reach the performance of ELMo(MIMIC) (Si et al., 2019), which is purely trained on clinical reports, making it more suited to the specific domain of the i2b2/VA 2010 clinical concept extraction task. Finally, and perhaps on an unrelated note, we can also see that the performance of ELMo(PubMed) is improved when this model is combined using the "mixture" method with static representations built from MIMIC. This goes to show that our methods can apply in a more general context whenever static embeddings can be expected to improve some pre-existing model, with a possible reason in this case being that the biomedical ELMo(PubMed) is probably missing some of the clinical knowledge that is available in the static MIMIC vectors.

4.4 Key Takeaways

All in all, we summarize our results in the following takeaways:

- Generally speaking, it seems that the larger the corpus the better. Similarly, the more domain-specific a corpus is, the more it is likely to result in improved performance.

- The question of whether a small specialized corpus performs better than a large one from the general-domain is still open. However, for a specialized corpus that is small enough (e.g. I2B2), large general-domain corpora seem to be a better option (recall I2B2 vs. GIGAWORD and WIKIPEDIA).
- All general-domain corpora are not made equal: depending on which subdomains are mixed within, some corpora may be more appropriate to specific target tasks than others (recall WIKIPEDIA which performs better than GIGAWORD).
- When dealing with in-domain corpora that are already quite large, domain similarity seems to become more important than additional size (recall PUBMED vs. MIMIC).
- In low resource scenarios, a small corpus consisting of texts from the target task can be leveraged in a way that achieves results that are on par with large in-domain corpora.
- Static representations trained on the target task corpus can be used to improve existing embeddings from the general domain, both static and contextual.
- Meta-embeddings through means of concatenation or linear combination (recall the "mixture" strategies) seem to be effective for combining in-domain and general representations.
- Neural language models may be further pre-trained on a small target task corpus to achieve seemingly systematic improvements in the target domain.
- While fine-tuned language models and in-domain static representations separately perform better than their general-domain analogs, both can be combined to achieve even greater improvements.

4.5 Conclusion

Word embeddings are an important component of modern NLP systems and corpora are probably the cornerstone of building good quality representations. In this chapter, we investigated some of the parameters that drive this quality, in particular, corpus size and domain similarity, in an attempt to shed some light on how these parameters interact to impact downstream performance. In the context of the medical domain, and leveraging a clinical sequence labeling task from I2B2/VA 2010, we evaluated multiple systems that relied on representations built from various corpora and demonstrated that both corpus size and domain similarity matter:

the former being perhaps more important when building a corpus from scratch, and the latter becoming more crucial when sufficient size has already been achieved. However, it is unlikely to have access to such corpora in practice, especially when dealing with specialized domains where both cultural and regulatory reasons may not allow sharing large amounts of in-domain texts. Therefore, we proposed methods leveraging the target-task corpus, an in-domain resource which we assume to be always available, for building better systems than otherwise achievable in such low-resource settings. These methods try to make the most of this task corpus by utilizing it in two different ways: for building static representations on one hand, and for refining existing general-domain embeddings on another. Given these two sets of independently trained representations, we have shown in different situations that a combination of both embeddings can lead to improved results that are often on par with systems leveraging large in-domain corpora. More specifically, we showed that pre-trained language models from the general domain (i.e. ELMo) could be further pre-trained on the target task corpus, then combined with static representations trained on the same corpus through either concatenation, or by incorporating the static vectors within the model architecture through linear combination, in order to achieve noticeable improvements. All in all, these methods tackle the low resource issue by optimizing the exploitation of available resources, leveraging both the small amount of in-domain information and open-source general-domain models. It is however conceivable to compensate for the lack of text resources with other kinds of structured data, more specifically, knowledge bases, which we explore in future parts of this manuscript (see Chapter 6). Finally, and in an effort to facilitate future research, we share our code along with instructions to reproduce our embedding strategies[16].

[16] https://github.com/helboukkouri/acl_srw_2019

Chapter 5

BERT and the WordPiece Vocabulary

5.1 Introduction

Before we tackle the topic of enhancing word representations using external knowledge, we take a quick detour to discuss BERT and BERT-like models as these have become the standard way to encode textual information in the NLP community. BERT is a language model that relies on a series of TRANSFORMER layers to contextualize input texts using a multi-headed self-attention mechanism. This model differs from previous language models such as ELMO in that it is significantly deeper (i.e. 12 or 24 layers for BERT vs. only 2 for ELMO), relies on transformers instead of recurrent layers, and is pre-trained on different language modeling tasks: Masked Language Modeling (MLM), where random words are masked and need to be recovered, and a self-explanatory Next Sentence Prediction task. Another key aspect of BERT is that it relies on a WordPiece system (Wu et al., 2016) where the model is assigned a vocabulary of subwords[1]. When a token is out-of-vocabulary, this token is recursively decomposed according to these subwords, converting it into a sequence of known elements. This sequence is then fed to the model which produces a representation for each subword. In the context of specialized domains, BERT models are often re-trained on large in-domain corpora in order to specialize their hidden representation for a target domain. However, this re-training does not affect the WordPiece vocabulary which remains constant and may drift away from the new domain. In the next section (§5.2), we discuss this issue and explore the impact this may have on downstream performance in a specialized setting by comparing models that rely on either a general or an in-domain WordPiece vocabulary. Following that (§5.3), we pro-

[1] A subword may be a full word but can also be a smaller part of a word or a token.

pose CharacterBERT, a variant of BERT that does not rely on a subword system but leverages instead a similar character-based system as ELMo in order to represent any incoming token as a single unit. We conduct an evaluation of CharacterBERT and train parallel versions of both BERT and its character variant in an effort to provide a fair comparison of both architectures. The results show that CharacterBERT overall improves over its BERT counterparts while at the same time displaying signs of improved robustness to noise and misspellings. All in all, we demonstrate that a subword system is not necessary for building transformer-based models that achieve good performance. Finally, we open-source our fine-tuning and pre-training codes as well as pre-trained models to enable future experiments on related topics[2,3,4].

While this chapter is not clearly set in a Domain Adaptation context, it nevertheless tackles topics that complement our previous investigations about corpus size and domain similarity, extending the analysis to a new hyperparameter which may affect the quality of specialized models: the WordPiece vocabulary—specifically, its domain. In fact, while our previous observations about corpus size and similarity do probably still hold in the context of BERT-like models, we assume here that we have access to corpora which are sufficiently large and domain-specific to allow for training good models. Given that, we explore how the new WordPiece vocabulary parameter affects the quality of this training and ultimately propose a variant of BERT that eliminates this parameter altogether by relying on character-based representations instead.

5.2 General Vocabulary, Specialized Domain

As mentioned, BERT and BERT-like models such as RoBERTa (Liu et al., 2019), XLNet (Yang et al., 2019) or ALBERT (Lan et al., 2020), rely on a tokenization method that leverages subwords in order to keep the vocabulary small and handle potential out-of-vocabulary tokens by decomposing them into a sequence of known WordPieces. This allows to strike a good balance between the efficiency of full words and the flexibility of characters, especially when constructing a model for a domain that is known in advance such as the default general domain. In fact, this WordPiece vocabulary is generally learned using a variant of BPE[5] (Gage, 1994), a compression algorithm which was adapted for tokenization purposes (Sennrich et al., 2016) such that the most frequent symbols are iteratively merged—forming

[2] https://github.com/helboukkouri/recital_2020

[3] https://github.com/helboukkouri/character-bert

[4] https://github.com/helboukkouri/character-bert-pretraining

[5] The exact implementation used for BERT seems to be internal to Google. However, most subsequent iterations like RoBERTa rely on the supposedly comparable BPE algorithm (see https://github.com/google/sentencepiece#comparisons-with-other-implementations).

character bi-grams, tri-grams, etc—and added to a subword vocabulary until a target size is reached. As a result, the output vocabulary is highly dependent on the corpus it was trained on—more specifically, its domain—which may entail some issues when the downstream applications cannot be expected to remain within the same original domain. In practice, the subword vocabulary is learned using the exact same corpus that is used for pre-training, which ensures this vocabulary is a perfect match for the pre-training domain. However, a mismatch occurs when the pre-trained model is used on a task from a different domain, which, given the cost of training domain-specific versions of such large models, is likely in practice. Furthermore, when the resources for training such models for the target domain are available, the most popular approach seems to be re-training existing general-domain systems on specialized corpora (e.g. SciBERT[6] (Beltagy et al., 2019), ClinicalBERT (Alsentzer et al., 2019)), probably in an attempt to leverage pre-existing knowledge within the model and speeding up convergence. As a result, these models also re-use the original subword vocabulary, which could affect the pre-training procedure in a way that may be harmful and lead to subpar models. In this section, we explore the effect of using a general-domain WordPiece vocabulary in the medical domain, focusing specifically on two different aspects: the quality of the tokenization and the downstream performance[7].

5.2.1 Effect on the Tokenization

In order to study the effect of using general-domain vocabularies in specialized domains, we first look into the tokenization results. Here, we assume that we have access to the original BERT model—which uses a general-domain WordPiece vocabulary—and investigate how this general vocabulary holds against specialized texts in the medical domain. Naturally, in order to have a reference to compare to, we first need to build our own domain-specific WordPiece vocabulary.

Building a Medical WordPiece Vocabulary

Domain	Corpora	# of documents	# of words
Medical	MIMIC-III	4.17 million	0.5 billion
	PMC OA abstracts	4.65 million	0.5 billion

Table 5.1: Corpus used for training a medical WordPiece vocabulary.

[6]There are versions of SciBERT that use a custom scientific vocabulary as well.
[7]This work has led to a publication at RECITAL 2020: (El Boukkouri, 2020)

Given a medical corpus which we put together using biomedical article abstracts from PubMed Central Open Access (PMC OA)[8] and MIMIC-III clinical notes (see Table 5.1), we aim to construct a WordPiece vocabulary that will, allegedly, be more suited for our medical domain than the general purpose WordPiece vocabulary occurring in BERT. To do so, we rely on an open-source implementation of the BPE algorithm[9] and learn a vocabulary of around 30k domain-specific subwords using the hyperparameters provided by default[10].

Tokenization Results & Discussion

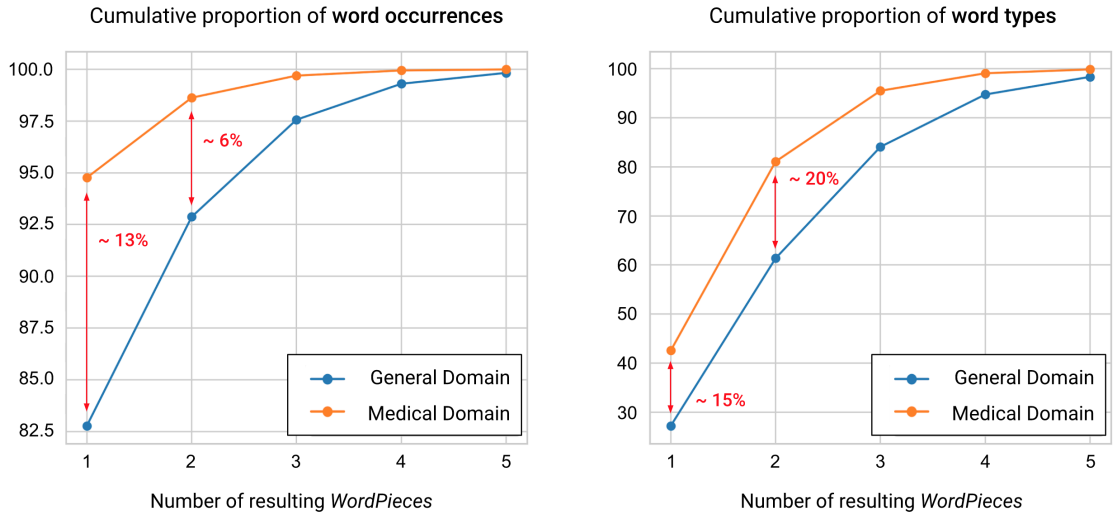

Figure 5.1: Number of subwords resulting from the tokenization of *a sample* of a medical corpus (see Table 5.1) using different domain WordPiece vocabularies.

After learning a medical WordPiece vocabulary, we randomly select a large sample ($\sim$ 1 million tokens) from the same medical corpus and tokenize it using either BERT's original vocabulary (general domain) or our custom specialized vocabulary (medical domain). Figure 5.1 shows the cumulative proportion of word types (i.e. distinct tokens) and word occurrences (i.e. counting each token as many times as they occur) against the number of resulting subwords after tokenization.

More Splits, More Often We notice a clear difference between the tokenization resulting from a general vocabulary and the one resulting from a specialized

[8] Available at: https://www.ncbi.nlm.nih.gov/pmc/tools/openftlist/
[9] Available at: https://github.com/google/sentencepiece
[10] See: https://gist.github.com/helboukkouri/374b9c74b8ee80a388f9d3943ebd245b

vocabulary. We note, for instance, that the medical vocabulary already includes $\sim$ 95% of word occurrences as full words (i.e. 1 WordPiece) against a lower $\sim$ 82.5% for BERT's default vocabulary. This means that around 12% of word occurrences are in-vocabulary for the medical set and out-of-vocabulary for the general-domain vocabulary. While this gap naturally tightens as the general and medical vocabularies approach the 100% coverage mark, we can see nevertheless that the medical WordPiece vocabulary leads overall to shorter subword sequences. An analysis of word type statistics is consistent with these observations, with, for instance, $\sim$ 60% of distinct token types resulting in two or less subwords when using a general vocabulary against $\sim$ 80% for the custom medical vocabulary.

Token	**Medical Tokenization**	**General Tokenization**
paracetamol	[paracetamol]	[para, ce, tam, ol]
choledocholithiasis	[choledoch, olithiasis]	[cho, led, och, oli, thi, asi, s]
borborygmi	[bor, bor, yg, mi]	[bo, rb, ory, gm, i]

Table 5.2: Tokenization of medical terms using different domain vocabularies.

Relatively Meaningless Subwords While having a subword vocabulary that systematically yields longer WordPiece sequences can be inconvenient, it is however unclear how this would negatively affect the final model. However, if we look more closely at the tokenization of some medical terms, we can notice that BERT's default vocabulary also leads to tokenizations involving less meaningful subwords. In fact, looking at Table 5.2, we can see that a common drug, *paracetamol*, is recognized by the medical vocabulary as a full word while on the other hand the general tokenization yields a sequence of subwords: *[para, ce, tam, ol]*. While in this situation, the prefix *para* is a common occurrence in everyday language for which we may assume that BERT will be able to learn a useful representation[11], the other subwords do not seem to represent any meaningful concept and therefore, would probably be assigned somewhat meaningless embeddings[12]. There are other examples that further illustrate the disparity between the medical and general vocabularies. For instance, we see that the medical vocabulary splits *choledocholithiasis*, a term which refers to the presence of gallstones in the common bile duct, into two subwords: *choledoch*, an actual medical term that refers to the common bile duct, and *olithiasis*, a common suffix that can be translated as "condition or presence of stones".[13] On the other hand, the general-domain vocabulary

[11]This is perhaps also the case for the suffix *ol* that can be common in the context of chemistry.

[12]Of course, what we may assume to be meaningless can still be useful for a deep neural network in ways that are not necessarily intuitive at first glance.

[13]See: https://clinicalanatomy.com/mtd-article-list/214-olithiasis

splits this medical term into a sequence of seemingly random subwords: *[cho, led, och, oli, thi, asi, s]*. Finally, we notice that for rarer terms like *borborygmi* (i.e. rumbling noises in the bowels), the medical tokenization includes less meaningful subwords as well (e.g. *bor*, *yg*) but still manages to detect a common suffix, *mi*, which occurs in the plural of some terms ending in *mus*. The general vocabulary, however, produces yet again a sequence of seemingly meaningless subwords.

Summary

Given our observations so far, it seems that a medical vocabulary might be more suited for training a domain-specific language model as it would result in less out-of-vocabulary words and a tokenization that generates *a priori* more meaningful subwords. These subwords would therefore be more likely to be assigned useful representations which can in turn facilitate dealing with any future unknown words. However, due to the nature of transformer-based models like BERT—and deep neural networks in general—it is possible that having seemingly arbitrary subwords may not affect the downstream performance as much as we would intuitively expect, as more relevant representations could be recovered within the model through successive contextualizations regardless of the information which we assume to be initially encoded within the static subword vectors. Therefore, further quantitative analysis is necessary in order to determine in practice whether using general-domain vocabularies in a specialized context may have as much of a negative impact as we might expect.

5.2.2 Effect on the Downstream Performance

The way domain-specific versions of BERT are constructed usually follows a simple methodology: first, the pre-trained weights from the original general-domain model are used to initialize a new model, then, the pre-training is resumed on a large in-domain corpus, ultimately leading to a specialized version of the model. As mentioned previously, this implies keeping the original general-purpose WordPiece vocabulary which, given that specialized vocabularies seem to overall produce a better tokenization, could lead to biases that may interfere with the domain-specific pre-training and in turn lead to overall weaker models. In this section, we investigate this idea by running a series of experiments where we compare the usual pre-training procedure for specialized models (i.e. general-domain initialization + re-training) against simply training new models from scratch using a domain-specific vocabulary. After training these models, we conduct an evaluation on multiple tasks from the medical domain to check our hypothesis and see to what extent, if any, the usual re-training method performs worse than training models from the ground up using in-domain vocabularies.

Model Configurations

Domain	Corpora	# of documents	# of words
General	WIKIPEDIA (EN)	11.9 million	2.14 billion
	OPENWEBTEXT	3.15 million	1.28 billion
Medical	MIMIC-III	4.17 million	0.5 billion
	PMC OA abstracts	4.65 million	0.5 billion

Table 5.3: Pre-training corpora from the general and medical domains.

We train a few different models using corpora from both the general and medical domains (see Table 5.3). More specifically, we rely on the BERT(base, uncased) architecture[14], which expects lowercased texts, consists of $L = 12$ TRANSFORMER layers with $H = 12$ attention heads each, and generates 768-dimensional contextual embeddings. We denote each model configuration according to three hyperparameters, namely: $\mathbf{V}$, the vocabulary domain; $\mathbf{C}_1$ the pre-training corpus; $\mathbf{C}_2$, a situational second pre-training corpus. The model configurations are:

($\mathrm{V} =$ **general,** $\mathrm{C}_1 =$ **general,** $\mathrm{C}_2 = \varnothing$) This is a BERT model that is trained on a general corpus and uses a general vocabulary. Training this model may seem redundant since there are already general-domain versions of BERT that are publicly available (e.g. original BERT). However, we make it a point to train all our models ourselves to ensure that their pre-training conditions are identical—or at least, as similar as possible—and therefore, that we can perform a fair comparison and isolate the effects of the parameters we chose to vary. Nevertheless, in an attempt to remain faithful to original BERT, we keep BERT's WordPiece vocabulary and compile a general-domain corpus that is the same size as the one used in the original paper (Devlin et al., 2019). Moreover, we attempt to have a similar corpus composition as well by using the entire English WIKIPEDIA. However, we are forced to replace the BOOKSCORPUS (Zhu et al., 2015) due to it not being publicly available. Instead, we use OPENWEBTEXT (Gokaslan and Cohen, 2019), an open-source reproduction or the WEBTEXT corpus[15] which relies on Reddit[16] upvotes to filter through the large amount of web data.

($\mathrm{V} =$ **general,** $\mathrm{C}_1 =$ **general,** $\mathrm{C}_2 =$ **medical**) We aim here to simulate the way domain-specific versions are trained by re-training a general model on an

[14]See: https://github.com/google-research/bert#pre-trained-models
[15]See: https://en.wikipedia.org/wiki/GPT-2#Training
[16]See: https://en.wikipedia.org/wiki/Reddit

in-domain corpus. Specifically, we keep the general-domain vocabulary and resume the pre-training of (V = **general**, C_1 = **general**, C_2 = $\varnothing$) on a medical corpus made of MIMIC-III and PMC OA[17] article abstracts.

(V = **medical,** C_1 = **medical,** C_2 = $\varnothing$) Contrary to previous models, this version is purely trained on medical texts. Moreover, we also drop the general-domain vocabulary and use instead a version that was learned specifically for the medical domain using the BPE algorithm[18] on our medical corpus.

(V = **medical,** C_1 = **medical,** C_2 = **medical**) We also consider a version that re-trains the purely medical model a second time on the same medical corpus. While we could simply train the first model once and for a longer period, we consider this somewhat awkward variant in order to get something that is directly comparable to re-training a general model on a medical corpus, but that starts instead from a medical model using an in-domain vocabulary. In other words, we include this configuration in our analysis in an effort to eliminate any biases relative to the total number of parameter updates.

Pre-training Setup

This section provides more details about the pre-training setup that is used to train our language model configurations. We perform training on 16 Tesla V100-SXM2-16GB GPUs, using the implementation and hyperparameters provided by NVIDIA[19]. Similar to original BERT, each training procedure consists of:

Phase 1 A series of 3,519 parameter updates with an overall batch size of 8,192 and a maximum input size of 128. This phase uses a learning rate of 6.10^{-3} and takes around 17 hours;

Phase 2 A fewer 782 parameter updates with a smaller batch size of 4096 on longer inputs with maximum size 512. This uses a smaller learning rate of 4.10^{-3} and takes around 9.5 hours.

During pre-training, we use the LAMB optimizer (You et al., 2020) as this has been shown to speed up convergence for large language models. We also use a linear schedule where the learning rate grows linearly during a certain number of training updates—i.e. 1000 steps for phase 1 and 100 steps for phase 2—before reaching its desired value and decreasing linearly to zero. Finally, we use a weight decay of 0.01 as is often the case with these models to further regularize training.

[17] This is the same corpus we used earlier in our tokenization experiments.

[18] Recall that BERT uses a slightly different implementation that is internal to Google.

[19] We adapt the code from: https://github.com/NVIDIA/DeepLearningExamples/tree/67b7543feb566b4e2ec520475db411f518b713f9/PyTorch/LanguageModeling/BERT

Evaluation Tasks

After pre-training our language models, we run an evaluation on multiple tasks from the medical domain, using the models as text encoders in order to compare the quality of their output representations. We consider the same I2B2/VA 2010 sequence labeling task as before[20], however, we also include a number of additional tasks from both the clinical and biomedical domains:

Clinical Entity Recognition This is the same sequence labelling task from **I2B2**/VA 2010 which we have used in previous evaluations (§4.2.2). We recall that this task consists in extracting three types of medical entities: PROBLEM (e.g. "headache"), TREATMENT (e.g. "oxycodone") and TEST (e.g. "MRI"), and uses exact span F1 as a metric. Figure 5.2 provides an example of a labeled sequence from the I2B2/VA 2010 concept extraction task.

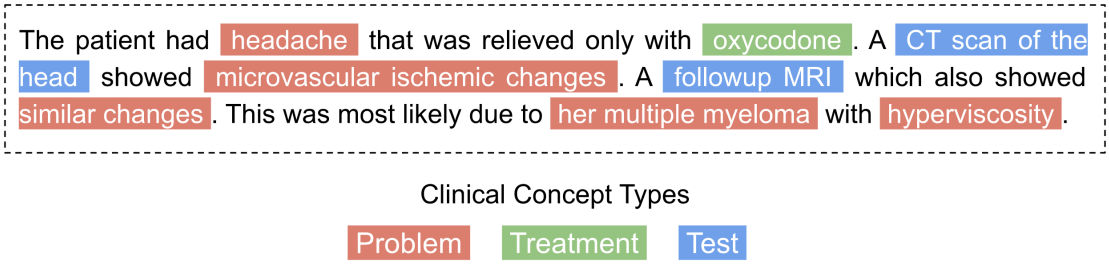

Figure 5.2: Example from the I2B2/VA 2010 clinical concept extraction task.

Natural Language Inference We also consider a new clinical Natural Language Inference task: **MEDNLI** (Romanov and Shivade, 2018). This task consists in classifying pairs of sentences into three categories: CONTRADICTION, when the second sentence contradicts the first one; ENTAILMENT, when the second sentence is implied by the first one; and NEUTRAL, when no clear relationship exists between both sentences. This task uses prediction accuracy as a metric. Figure 5.3 shows examples of labeled sentence pairs.

Relation Extraction In order to include a diverse range of tasks, we also evaluate our models on two additional biomedical relation extraction tasks: **DDI**,

[20]Please note that for the i2b2 task as well as for all subsequent sequence labeling tasks, when using a WordPiece-based model such as BERT, we evaluate at the level of the subword. Specifically, we tokenize the texts at the WordPiece-level, adapt the word-level gold standard accordingly by projecting the label of each word on its WordPieces, then predict a label for each subword. Ultimately, the models are required to recover each WordPiece label correctly and not simply the label of the first WordPiece of each token.

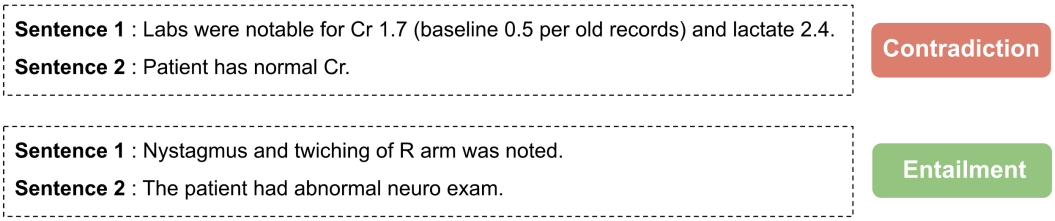

Figure 5.3: Examples from the MEDNLI task.

which is originally from SemEval 2013 - Task 9.2[21] and focuses on classifying drug-drug interactions into ADVISE (DDI-advise), EFFECT (DDI-effect), MECHANISM (DDI-mechanism), INTERACTION (DDI-int), and DDI-false for no interaction, as well as **CHEMPROT** , originally from BIOCREATIVE VI[22], which focuses on chemical-protein relations and requires them to be classified into ACTIVATOR (CPR:3), INHIBITOR (CPR:4), AGONIST (CPR:5), ANTAGONIST (CPR:6), SUBSTRATE (CPR:9) and FALSE for no relation. Both tasks use the micro-averaged F1 over non-negative classes as a metric. Figure 5.4 shows samples from both the CHEMPROT and DDI relation extraction tasks.

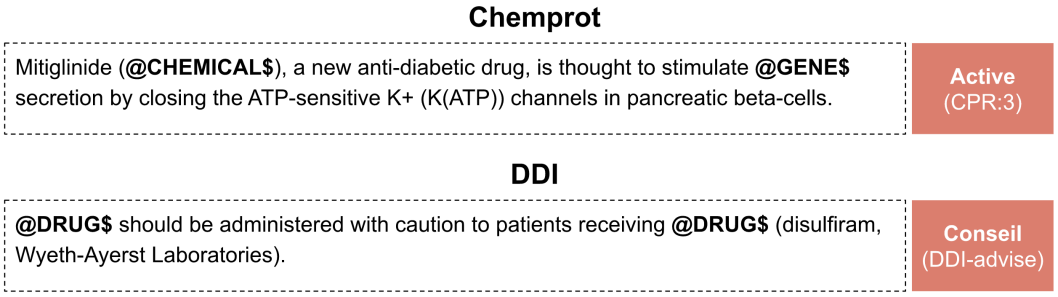

Figure 5.4: Examples from the CHEMPROT and DDI tasks.

For each evaluation task, we report in Table 5.4 the number of samples, as well as the number of entities or positive relations when appropriate.

[21] See: https://www.cs.york.ac.uk/semeval-2013/task9/

[22] Available at: https://biocreative.bioinformatics.udel.edu/resources/corpora/chemprot-corpus-biocreative-vi/

	I2B2/VA 2010	MEDNLI	CHEMPROT	DDI
Training	24,757 (22,263)	11,232	19,460 (4,154)	18,779 (2,937)
Validation	6,189 (5,565)	1,395	11,820 (2,416)	7,244 (1,004)
Test	45,404 (45,009)	1,422	16,943 (3,458)	5,761 (979)

Table 5.4: Number of examples per evaluation task. We report the number of entities for I2B2 and positive relations for CHEMPROT and DDI between parentheses.

Evaluation Setup

For each evaluation task, we follow the common practice and add the usual task-specific layers on top of the pre-trained BERT model[23]. We then fine-tune these architectures for 15 epochs using a batch size of 32. After each epoch, we run an evaluation step on the provided validation set, using 20% of the training data for validation otherwise, and ultimately select the best performing checkpoint. Similar to previous experiments, we account for possible randomness issues by repeating each experiment using 10 different random seeds and computing a final score as mean ± standard deviation (see Chapter 3 on model evaluation).

Results and Discussion

We report the performance of all model configurations in Table 5.5. Moreover, we also provide a set of baselines by fine-tuning the original general-domain BERT (Devlin et al., 2019) as well as BLUEBERT (Peng et al., 2019), a publicly available medical version of BERT.

Pre-training Sanity Check Given the complexity involved in training large language models like BERT, it is useful to first compare the performance of our general-domain model with the original BERT. Looking at the results of these models, we can see that both have relatively similar performances with, nevertheless, an advantage for the original version of BERT. This may be explained by the fact that we use slightly different pre-training corpora (i.e. OPENWEBTEXT vs. BOOKSCORPUS) as well as different pre-training parameters (i.e. batch size, number of updates, learning rate)[24]. Moreover, this demonstrates the benefit of training multiple models in the same conditions, as this allows to eliminate such

[23]You can refer to Section 4 from the original paper (Devlin et al., 2019) for the authors' suggestions. Moreover, appendix B provides a number of illustrations including BERT-based architecture for entity recognition, NLI and classification. However, please note that we do not use a CRF layer in the current set of experiments.

[24]In fact, due to server limitations, we trained our models for half the number of steps suggested in NVIDIA's implementation. This could mean that our models are overall under-trained.

Model			Evaluation Task			
V	**C_1**	**C_2**	**I2B2**	**MEDNLI**	**CHEMPROT**	**DDI**
G	G	∅	85.66 ± 0.18	77.31 ± 0.71	67.47 ± 0.99	75.81 ± 1.02
G	G	M	89.00 ± 0.17	**84.91 ± 0.46**	72.29 ± 0.58	78.82 ± 1.11
M	M	∅	88.80 ± 0.10	83.54 ± 0.43	71.30 ± 0.51	79.40 ± 1.15
M	M	M	**89.20 ± 0.20**	84.32 ± 0.73	**72.97 ± 0.46**	**80.11 ± 0.79**
Baselines						
BERT			86.42 ± 0.31	77.85 ± 0.63	69.22 ± 0.56	77.89 ± 0.92
BLUEBERT			88.70 ± 0.21	84.53 ± 0.76	68.35 ± 0.61	77.89 ± 0.65

Table 5.5: Evaluation results across different tasks from the clinical (I2B2, MEDNLI) and biomedical (CHEMPROT , DDI) domains. The best performance is shown in **bold** and the second best is underlined. To save space, we denote the general domain as 'G' and medical domain as 'M'. BERT refers to the original BERT(base, uncased) and BLUEBERT refers to the model from (Peng et al., 2019).

initial differences in performance and provides a level playing field for all models. All in all, given that we are able to perform at close enough levels to the original BERT, we may assume that our implementation is correct and resume our analysis.

General BERT < Medical BERT As expected, we see that general models using general vocabularies (V = G, C_1 = G, C_2 = ∅) perform worse than specialized models using in-domain vocabularies (V = M, C_1 = M, C_2 = ∅). Moreover, this remains true for the original BERT (baseline) which is outperformed by our custom medical model as well as the medical baseline BLUEBERT. Incidentally, it seems that using a medical vocabulary and training a model directly on a medical corpus (V = M, C_1 = M, C_2 = ∅) leads to better results than BLUEBERT, which re-trains the original BERT on a similar medical corpus[25]. This could hint at using a specialized vocabulary and training directly on in-domain data as being possibly better than the plain re-training of a general model on a specialized corpus.

Is Training from Scratch Really Better? Despite the fact that our medical model performs better than BLUEBERT, a version that retrains the original BERT on a medical corpus, we see that training from scratch with a specialized

[25]Their corpus, however, includes 8 times as many biomedical texts as we do. Interestingly, we outperform this model on CHEMPROT and DDI, both of which are from the biomedical domain.

vocabulary is not necessarily better, since re-training our general model on a specialized corpus (V = G, C_1 = G, C_2 = M) eventually leads to better results. This reaffirms the importance of training models in similar conditions and warrants the inclusion of a medical model that is trained a second time on the same medical corpus. In fact, by re-training the purely medical model a second time (V = M, C_1 = M, C_2 = M), we manage to improve over the previously mentioned re-trained version on most evaluation tasks, with the exception of MEDNLI. This goes to show that the overall training time of these models does matter, and to a point where a purely specialized model may perform worse than a model relying on general-domain representations if the former is not trained properly.

Summary

Our results clearly demonstrate how important it is to train models in similar conditions before attempting to draw conclusions about which configuration might be better. In fact, by constructing multiple variants of BERT using either a general-domain or a medical vocabulary and relying on either a general or a specialized corpus, we are able to realize that while a purely medical model does perform better than a purely general version, re-training the general version on a specialized corpus does overall lead to results that are on par with a model using a tailored vocabulary with a large in-domain corpus. Moreover, we notice that besides the domain of the WordPiece vocabulary and the training corpus, the training time (i.e. overall number of parameter updates) plays an important role. In fact, we show that a purely medical model may perform below a general model that uses a general vocabulary and is retrained on a specialized corpus, especially if the former is not trained for as long of a time. All in all, given the cost required for training a purely specialized model twice, it seems preferable to re-train a general model as the latter approach seems to be overall comparable in terms of performance. This means that despite using a general-domain vocabulary that produces seemingly meaningless subwords during tokenization, BERT is able to alleviate these apparent issues, probably contextualizing these subwords in a way that enables recovering some of the original token semantics. In the end, the domain of the WordPiece vocabulary may or may not have an actual impact on downstream performance. However, relying on these subwords is definitely inconvenient in practice, leading for instance to multiple WordPiece embeddings instead of a single word representation. In the next section, we explore the idea of a BERT model that does not rely on such subwords, using instead a system that consults token characters.

5.3 Characters Instead of Subwords

Due to the compelling improvements brought by BERT, many recent representation models have adopted a similar approach, consequently inheriting the WordPiece tokenization system. While this subword system arguably achieves a good balance between the flexibility of characters and the efficiency of full words, using predefined vocabularies from the general domain may not always be suitable, especially when building models for specialized domains (e.g., the medical domain). Moreover, adopting a WordPiece tokenization shifts the focus from the word level to the subword level, making the models conceptually more complex and arguably less convenient in practice. In this section, we explore the idea of a subword-free transformer-based language model. Specifically, we would like to build and evaluate a variant of BERT that does not rely on the WordPiece system. This system would consequently avoid any biases that may arise from using pre-defined WordPiece vocabularies (e.g. a general vocabulary in a specialized domain) while at the same time representing a step towards reverting back to the conceptually simpler family of word-level models. In fact, there have been several NLP systems that achieved an open-vocabulary while representing full words (Luong and Manning, 2016; Kim et al., 2016b; Jozefowicz et al., 2016). One such system is the LSTM-based model ELMo which is able to produce a single contextual representation for any arbitrary token[26] by consulting its characters and without requiring any pre-defined word or subword vocabulary that could be domain-dependent. In this work, we propose CharacterBERT, a variant of BERT that uses ELMo's character-based system instead of WordPieces. By training parallel versions of both BERT and CharacterBERT, we investigate whether WordPieces are necessary for building good BERT-like language models and study whether a vocabulary-free model could positively impact transfer across different domains[27].

5.3.1 CharacterBERT

CharacterBERT uses the exact same architecture as vanilla BERT but relies on a different method for constructing its initial context-independent representations: while the original model consults a subword vocabulary to split unknown tokens into two or more WordPieces, then embeds each unit independently using a WordPiece embedding layer, CharacterBERT uses a Character-CNN module (Peters et al., 2018a; Jozefowicz et al., 2016) which, much like ELMo's initial layer, consults the characters of each token to produce single token representations (see Figure 5.5).

[26] As long as it is not unreasonably long (i.e. under 50 characters).

[27] This work has led to a publication at COLING 2020: (El Boukkouri et al., 2020)

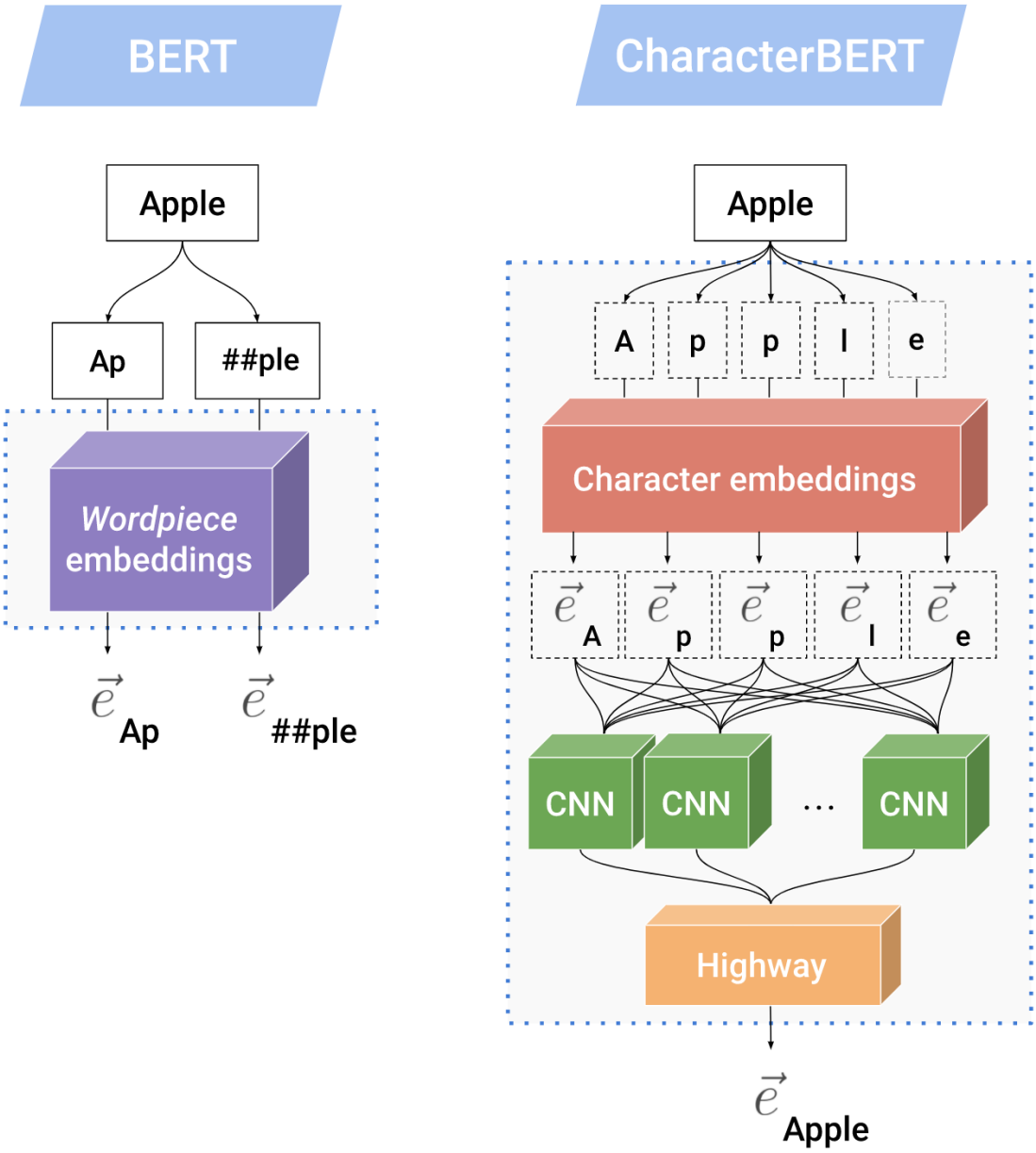

Figure 5.5: Comparison of the context-independent representation systems in BERT and CharacterBERT. In this illustration, BERT splits the word "Apple" into two WordPieces, then embeds each unit separately. CharacterBERT, on the other hand, produces a single vector for "Apple" by consulting its characters.

CharacterCNN: Building Word Representations from Characters

We use the same CharacterCNN architecture that is implemented as part of ELMo, adding support for some special symbols that are specific to BERT (e.g. [MASK], [CLS]). This character module constructs context-independent token representations by following a number of steps:

1. In simple terms, each token is seen as a sequence of characters. In practice, however, each character is replaced with the appropriate UTF-8 byte sequence, producing a long sequence of bytes for each input token which we rely on instead of the actual characters. This has the benefit to limit the vocabulary size to the total number of possible bytes (i.e. 256)[28]. Moreover, it allows for the possibility of using the same byte vocabulary for different kinds of inputs (e.g. different languages, non-ascii symbols).

2. Each byte sequence is truncated to a maximum size of 50. This rarely happens in practice and allows to pad all byte sequences to the same target size in order to feed multiple sequences in batches and speed up computation.

3. Using a "character embedding layer", each byte is converted into a 16-dimensional vector. Consequently, each input word becomes a sequence of 16-d character (byte) representations. Here, a vanilla BERT model would have split an out-of-vocabulary token into WordPieces, producing a sequence of subword representations. Then, this sequence would have been biased using positional and segment embeddings before being fed to a sequence of Transformer layers for contextualization. However, with CharacterBERT, the plain character embeddings cannot be leveraged directly since, for instance, the self-attention mechanism would not be sensitive to character order (e.g. "apple" and "palpe" would be seen as the same token) and the character representations may not encode sufficient information. As a result, additional processing steps are required to further digest the sequence of character vectors.

4. Each character embedding sequence is fed to multiple CNNs with different filters sizes in order to extract, essentially, signals at various n-gram levels. Specifically, we use the following [*filter size*, *number of filters*] configurations: [1, 32], [2, 32], [3, 64], [4, 128], [5, 256], [6, 512] and [7, 1024]. The output of each CNN is then max-pooled across the character sequence, which can be interpreted as filtering only the strongest signals for each n-gram. Finally, all outputs are concatenated to form a global feature vector for each token with dimensionality: $32 + 32 + 64 + 128 + 256 + 512 + 1024 = 2048$.

[28] Counting the special symbols such as [CLS] and [SEP], the final vocabulary size is 263.

5. Each feature vector goes through two successive Highway layers (Srivastava et al., 2015) which apply non-linearities with residual connections and eventually project these representations down to a final embedding size which we choose to match BERT's 768-dimensional output[29]. Ultimately, the final output for each token is a single context-independent vector that is supposed to play a similar role to BERT's WordPiece representations.

As with BERT, we add position and segment encodings to the aforementioned token embeddings—i.e the CharacterCNN vectors—before feeding the resulting representations to the downstream Transformer layers for contextualization. Since CharacterBERT does not split out-of-vocabulary tokens into WordPieces, each input token goes through the system as a single vector, ultimately leading to a single contextual representation.

5.3.2 Experiments

In order to evaluate the impact of using a CharacterCNN instead of the usual WordPiece embeddings, we compare BERT and CharacterBERT on several medical tasks. In a similar spirit to previous pre-training experiments, we train each CharacterBERT model alongside a BERT counterpart, in the exact same settings, in order to dissociate the impact of the CharacterCNN from other effects related to model training (e.g. training corpora and hyperparameters).

Model Configurations

We train multiple models following the BERT(base, uncased) architecture which expects lowercase texts and uses 12 Transformer layers with 12 attention heads, producing 768-dimensional representations. This architecture has $\approx$ 109.5M parameters, which become in the case of CharacterBERT $\approx$ 104.6M parameters. It is interesting to note that using a CharacterCNN actually results in an overall smaller model despite using a character module which may look complex at first glance. This may be explained by the fact that BERT's WordPiece embeddings matrix has $\approx$ 30K $\times$ 768-d vectors while CharacterBERT uses smaller 16-d character embeddings and mostly small-sized CNNs.

For each architecture, we consider two different model configurations: one general and one specialized (i.e. medical). The specialized versions are re-trained from the general models in accordance with the usual way in which BERT is adapted to new domains. We use the following model configurations:

[29] Assuming we are using the BERT(base, uncased) architecture.

BERT$_{\text{general}}$ This is a general-domain model obtained by pre-training BERT on a general corpus. It uses the same architecture and WordPiece vocabulary as BERT(base, uncased), which it is meant to replicate.

BERT$_{\text{medical}}$ This is a medical version of BERT that is the result of re-training BERT$_{\text{general}}$ on a large medical corpus.

CharacterBERT$_{\text{general}}$ This is a general-domain model obtained by training CharacterBERT on a general corpus. Besides the CharacterCNN, it uses the same architecture as BERT$_{\text{general}}$, which it is meant to replicate with a different tokenization and representation approach.

CharacterBERT$_{\text{medical}}$ This is a medical version of CharacterBERT that is the result of re-training CharacterBERT$_{\text{general}}$ on a large medical corpus. This is supposed to be the CharacterCNN analog of BERT$_{\text{medical}}$.

Pre-training Setup

We use the same general and medical corpora as with previous pre-training experiments (see §5.2.2, Table 5.3). We recall that the general corpus is slightly different from BERT's original corpus due to replacing BooksCorpus with OpenWebText and that the medical corpus is comprised of clinical notes from MIMIC-III and biomedical article abstracts from PMC-OA. Following the common practice for BERT, we pre-train our models on two different tasks: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). BERT and CharacterBERT are pre-trained identically. However, during Masked Language Modeling we mask and predict entire tokens with CharacterBERT instead of the usual subwords that are masked when using BERT. This is a natural consequence of having a word-level model and can be seen as enforcing Whole Word Masking for free, which, incidentally, has been shown to improve the quality of vanilla BERT models[30] (Cui et al., 2019). We pre-train all model configurations following the exact same procedure described in Section 5.2.2.

Evaluation Tasks

We evaluate our model configurations on a wide range of tasks which includes previously presented evaluations tasks (§5.2.2) as well as an additional sentence similarity task, always in an attempt to consider a diverse range evaluation setups.

[30]Google has updated their model repository with Whole Word Masking versions which were shown to improve over the original BERT. See: https://github.com/google-research/bert

Medical Entity Recognition We reuse the I2B2/VA 2010 clinical concept extraction task which aims to extract three types of medical concepts: PROBLEM (e.g. "headache"), TREATMENT (e.g. "oxycodone"), and TEST (e.g. "MRI"). This uses exact span F1 as a metric.

Natural Language Inference We also reuse the clinical natural language inference task MEDNLI, which aims to classify sentence pairs into three categories: CONTRADICTION, ENTAILMENT, and NEUTRAL. This uses prediction accuracy as a metric.

Relation Classification We reuse the two biomedical relation classification tasks as well, namely: CHEMPROT and DDI, which respectively focus on classifying chemical-protein interactions: ACTIVATOR (CPR:3), INHIBITOR (CPR:4), AGONIST (CPR:5), ANTAGONIST (CPR:6), or SUBSTRATE (CPR:9); and drug-drug interactions: ADVISE (DDI-advise), EFFECT (DDI-effect), MECHANISM (DDI-mechanism), or INTERACTION (DDI-int). These tasks use a micro-averaged F1 on positive classes as a metric.

Sentence Similarity Finally, we include an additional sentence similarity task from the clinical domain: CLINICALSTS (Wang et al., 2020). This is a task from the BIOCREATIVE/OHNLP Challenge 2018, Task 2 (Wang et al., 2018) which aims to produce similarity scores for sentence pairs that correlate with the gold standard. As a similarity task, this uses Pearson correlation as a metric.

Figure 5.6 provides examples from each evaluation task and Table 5.6 reports the number of samples, entities and positive relations for these same tasks.

	I2B2	MEDNLI	CHEMPROT	DDI	CLINICALSTS
Train.	24,757 (22,263)	11,232	19,460 (4,154)	18,779 (2,937)	600
Val.	6,189 (5,565)	1,395	11,820 (2,416)	7,244 (1,004)	150
Test	45,404 (45,009)	1,422	16,943 (3,458)	5,761 (979)	318

Table 5.6: Number of examples per evaluation task. We report the number of entities for I2B2 and positive relations for CHEMPROT and DDI between parenthesis.

Evaluation Setup

We take the same approach to evaluating our models as with previous experiments. Namely, given a pre-trained model, and evaluation task and a seed $i \in 1..10$:

1. We fine-tune a task-specific architecture for 15 epochs, using batches of 32.

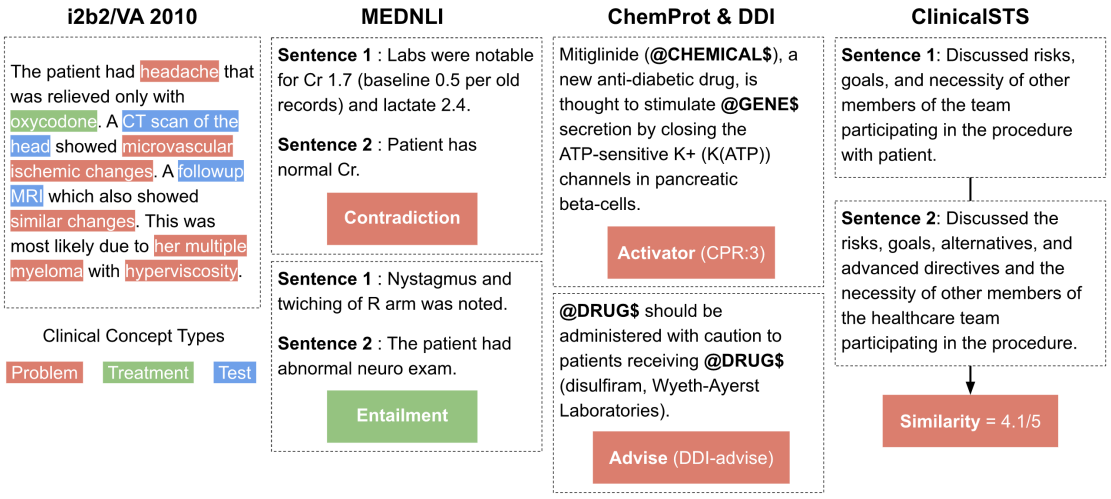

Figure 5.6: Examples from each evaluation task.

2. After each epoch, we evaluate the model on a validation set that is either given or computed as 20% of the training set. According to the validation performance, we may save the model.

3. After completion, we load and evaluate the best model on the test set.

This process is repeated for all seeds to compute a final performance as *mean* ± *std*. We have mentioned previously how this allows to account for some of the variability during training. However, this time we also use the multiple seeds to build ensembles as detailed in Section 3.3. These ensembles are eventually used to compute a final score as *mean* ± *std*. All fine-tuning experiments are run on a single Tesla V100-PCIE-32GB using Adam (Kingma and Ba, 2015) with a learning rate of 3e-5, a warm-up ratio of 10%, and a weight decay of 0.1.

5.3.3 Results & Discussion

Speed Benchmark

Pre-training Pre-training a single model through phases 1 and 2 using our hyperparameters takes around 26.5 hours for BERT and 55 hours for CharacterBERT, despite both architectures having about the same number of parameters. This great difference in pre-training speed is partly due to the CharacterCNN being a bit slower to train, being more complex than the original WordPiece embedding matrix (i.e. multiple CNNs and Highway layers). However, the main reason for the pre-training being slower is probably that we are not able to use

a very specific trick during Masked Language Modelling which BERT is able to use to speed up computation. In fact, when using BERT, all input tokens are elements of the WordPiece vocabulary given that the tokenization step splits any new unknown token into a sequence of known subwords from said vocabulary. As a result, when performing MLM predictions, recovering the masked symbol is the same as selecting the most likely element from the WordPiece vocabulary. Consequently, the output and input vocabularies are the same and BERT is able to share its WordPiece embedding matrix with the output layer, which tends to speed up training. On the other hand, with CharacterBERT, a consequence of dropping the WordPiece system is that our model does not have a subword embedding matrix anymore and that the input text is tokenized into full words. Since the input token representations are generated on the fly using the CharacterCNN, it is not straightforward how to re-use the same system to recover masked tokens during Masked Language Modeling. As a result, we are compelled to build a temporary output vocabulary which we limit to the top 100K tokens in the training corpus and use as MLM targets. One direct consequence of this is that only symbols from the 100K vocabulary are allowed to be masked, which may limit the assimilation of rarer words and expressions. While there might be a clever way to invert the CharacterCNN in order to use it for generating MLM predictions and share it with the output layer similar to what is done with vanilla BERT, it is unclear how that would affect the quality of the final models. One possible solution that might be worth exploring is using methods like Noise Contrastive Estimation (Mnih and Kavukcuoglu, 2013) to iteratively contrast more or less likely tokens instead of the costly discriminative selection of specific elements from a large vocabulary. These methods, however, are left for future work.

Fine-tuning In addition to the pre-training speed, we also report the elapsed time for fine-tuning as well as for inference. Figure 5.7 shows that CharacterBERT is much less at a disadvantage when it comes to fine-tuning (19% slower on average instead of 108%). However, in the specific case of the DDI task, we can see that CharacterBERT is actually 14% faster than BERT. While surprising at first, we notice that the DDI texts include an important number of complex terms which are not part of BERT's original vocabulary (e.g. cholestyramine, which becomes [cho, les, ty, ram, ine]), which leads to an increased average input length for BERT relative to the vocabulary-free alternative, CharacterBERT. For instance, the minimum, average and maximum input lengths from the DDI test set are respectively 4, 34, and 94 for CharacterBERT against 9, 79, and 169 for BERT. Given the quadratic complexity of these BERT-like models with respect to their input length, this could explain why CharacterBERT is faster during the fine-tuning on DDI task. At inference time, we can see that Charac-

TERBERT is overall faster than vanilla BERT, as the CHARACTERCNN is not as slow during inference as it is during optimization. All in all, there seems to be an interplay between, on one hand, the word-level tokenization which leads to shorter input sequences and makes CHARACTERBERT faster, and, on the other hand, the complexity of the CHARACTERCNN module which slows down the model in situations where back-propagation is applied.

Fine-tuning (w/ Tesla V100-PCIE-32GB)

	i2b2	MEDNLI	ClinicalSTS	DDI	ChemProt
BERT	3:36:20	1:09:29	0:02:58	1:32:42	2:42:36
CharacterBERT	4:29:01	1:22:40	0:04:12	1:19:43	3:25:31
Relative difference	**+24.35%**	**+18.97%**	**+41.57%**	**-14.01%**	**+26.39%**

Inference (w/ Tesla V100-PCIE-32GB)

	i2b2	MEDNLI	ClinicalSTS	DDI	ChemProt
BERT	0:11:16	0:00:11	0:00:01	0:00:31	0:02:28
CharacterBERT	0:10:57	0:00:10	0:00:01	0:00:22	0:02:44
Relative difference	**-2.81%**	**-9.09%**	**0%**	**-29.03%**	**+10.81%**

Figure 5.7: Speed comparison during fine-tuning and inference.

Preliminary Analysis

Reproducing Vanilla Models We report the performance of BERT(base, uncased) as well as BLUEBERT(base, uncased) (Peng et al., 2019), a medical model pre-trained on MIMIC-III and PUBMED abstracts[31]. As with previous experiments, we include these two baselines in order to compare our custom general and medical BERT to external models trained by the community and evaluate the quality of our pre-training procedure. Note that $\text{BERT}_{\text{general}}$ and $\text{BERT}_{\text{medical}}$ are the same two models that use a general vocabulary and which we have presented in Section 5.2.2. Figure 5.8 shows that $\text{BERT}_{\text{general}}$ performs slightly worse

[31]Note that BLUEBERT is trained on PUBMED abstracts while our medical models are trained on the smaller set of PMC OA abstracts.

than the original BERT despite using exactly the same architecture. We have commented on this previously and we now validate this result using the ensemble scores and on an additional sentence similarity task. As mentioned previously, this difference may be attributed to either the different general-domain corpora (OpenWebText instead of BooksCorpus) or to differences in pre-training parameters (number of updates, batch size...). Moreover, we see that $\text{BERT}_{\text{medical}}$ performs at the same level as BlueBERT, sometimes outperforming the latter substantially ($\approx$ +4 F1 on ChemProt), which allows us to safely assume the proper functioning of our pre-training procedure.

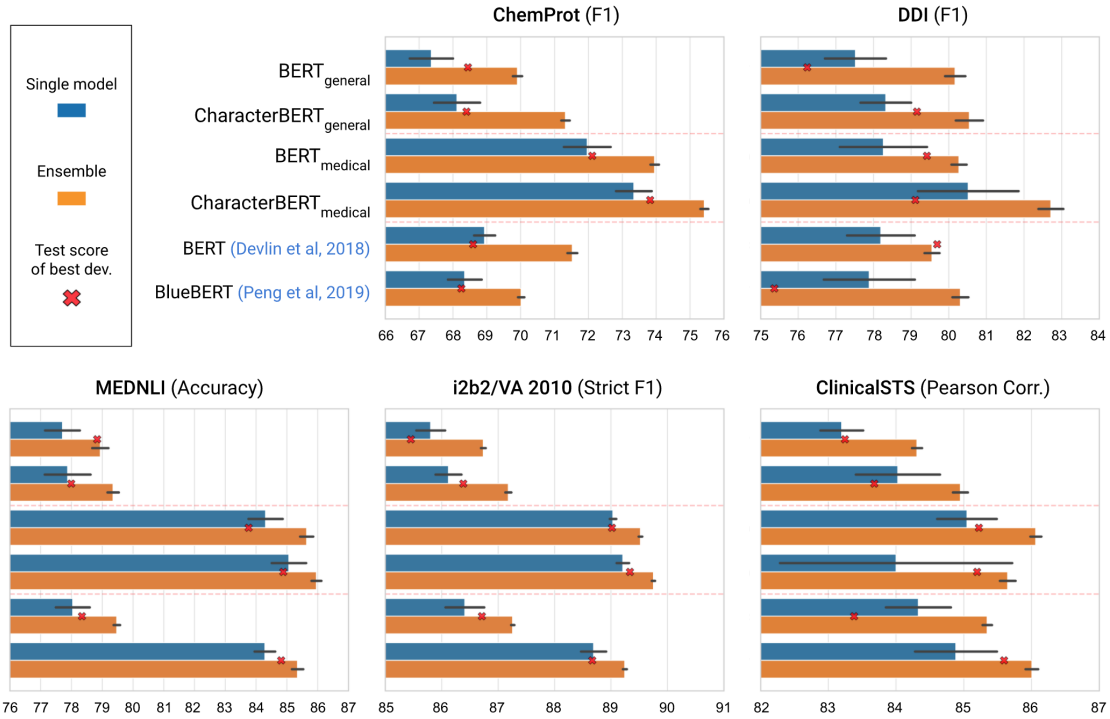

Figure 5.8: Comparison of pre-trained models when fine-tuned on several medical tasks. For each model, the test performance of 10 random seeds is expressed as *mean* $\pm$ *std* and is shown in **blue** for single models and **orange** for ensembles. The test set performance of the best validation seed is shown as a **red** symbol.

Ensembles and Model Selection Figure 5.8 shows the average ensemble performance (orange bars) as well as the average performance of single models (blue bars). One immediate observation is that ensembles systematically perform better than single models, sometimes with great improvements (e.g. $\sim$ +3 F1 for CharacterBERT$_{\text{general}}$ on ChemProt and $\sim$ +2 F1 for CharacterBERT$_{\text{medical}}$

on DDI). While not surprising *per se*, it is worth noting that these ensembles were produced using a simple majority voting strategy which can be easily and cheaply applied as a post-processing step. Moreover, if we assume the availability of multiple runs per experiment—which we argue is a always a good practice—one common approach is to select the best performing seed on the validation set and use that as a final model. Our results show that this model (shown as a red symbol) performs always worse than the average ensemble model. Additionally, we can see that model ensembles have substantially lower variances compared to single models, which, all in all, suggests that computing model ensembles is an easy and effective way to improve the overall performance while allowing for a more reliable comparison across different model configurations.

Detailed Test Scores Along with graphical representations of the results, we also provide a more detailed version of the test scores in Figure 5.9. For each model, either using the single model scores (i.e. row prepended with 'S') or the model ensembles (i.e. rows with 'E'), we compute the first, second (median), and third quartiles of the score distribution. We also use a blue-red color gradient to highlight for each column the values at both ends of the spectrum (i.e. blue $<$ red). A quick look at the table shows a concentration of high scores for the medical BERT and CharacterBERT models, with the latter performing better for all tasks but ClinicalSTS. Incidentally, while the bar plot for ClinicalSTS (Figure 5.8) shows a great variance for single $\text{CharacterBERT}_{\text{medical}}$ models, the detailed quartile values give some insights on where this variance comes from. In fact, we can see that the first quartile of the single model performance is very low with a correlation of 82.92 while medical BERT achieves a higher value of 84.80. However, we also see that the difference is smaller for the median and third quartiles with respective deltas of only 0.18 and 0.05. Therefore, CharacterBERT's performance on ClinicalSTS may be explained with the presence of a few "bad seeds" where the model does not manage to converge properly. Such seeds would have a substantially worse performance and would only count in the computation of the first quartile of the score distribution, explaining the large variance and the mismatch between the first and other quartiles[32].

BERT Vs. CharacterBERT: How Significant Is the Difference?

Our evaluation results show that, overall, we are able to outperform original BERT using the CharacterBERT and without relying on any kind of subword system. This is evidenced by the visual color coding in Figure 5.9, as well as

[32]To be clear, these "bad seeds" are probably not "unlucky seeds" but rather a manifestation of a heavier tail at the lower end of the score distribution.

		ChemProt (F1 score)			DDI (F1 score)			MEDNLI (Accuracy)			i2b2/VA 2010 (Strict F1 score)			ClinicalSTS (Pearson Correlation)		
		Q1	Q2	Q3	Q1	Q2	Q3	Q1	Q2	Q3	Q1	Q2	Q3	Q1	Q2	Q3
$BERT_{general}$	S	66.94	67.04	67.84	76.93	77.40	77.99	77.43	77.67	77.94	85.72	85.86	85.97	83.07	83.22	83.35
	E	69.81	69.88	69.98	79.99	80.14	80.35	78.71	78.90	79.15	86.71	86.73	86.77	84.27	84.33	84.36
$CharacterBERT_{general}$	S	67.89	68.24	68.38	77.89	78.17	79.03	77.18	78.09	78.50	85.99	86.18	86.28	83.63	83.91	84.09
	E	71.27	71.30	71.40	80.39	80.54	80.88	79.20	79.29	79.47	87.15	87.17	87.23	84.94	84.97	85.03
$BERT_{medical}$	S	71.76	71.93	72.23	77.54	77.93	79.15	83.79	84.25	84.69	88.99	89.01	89.08	84.80	84.98	85.20
	E	73.85	73.94	74.01	80.14	80.20	80.38	85.51	85.65	85.78	89.49	89.51	89.55	86.01	86.08	86.12
$CharacterBERT_{medical}$	S	72.84	73.44	73.78	79.18	80.38	81.70	84.56	84.95	85.58	89.14	89.24	89.30	82.92	84.80	85.15
	E	75.31	75.40	75.50	82.44	82.74	83.01	85.83	85.97	86.11	89.73	89.75	89.77	85.54	85.62	85.76
BERT (Devlin et al, 2018)	S	68.67	68.82	69.18	77.67	78.08	78.83	77.67	78.02	78.29	86.23	86.54	86.61	83.97	84.44	84.65
	E	71.46	71.54	71.64	79.46	79.49	79.61	79.41	79.47	79.54	87.23	87.26	87.28	85.32	85.37	85.40
BlueBERT (Peng et al, 2019)	S	68.25	68.31	68.69	77.55	77.89	78.74	84.07	84.25	84.55	88.47	88.73	88.87	84.39	84.98	85.39
	E	69.93	69.98	70.10	80.26	80.33	80.43	85.25	85.41	85.44	89.22	89.24	89.28	85.95	85.99	86.06

Figure 5.9: Performance of our pre-trained models when fine-tuned on five different medical tasks. Two baselines are included: BERT (Devlin et al., 2019) using the "base-uncased" architecture and BlueBERT (Peng et al., 2019), a medical BERT that is the result of re-training the former model on MIMIC-III and PubMed abstracts. Legend: Qi = i-th quartile, E = ensemble, S = single model, **blue** < **red**.

the bar plots from Figure 5.8 where we can see in particular that CharacterBERT$_{\text{medical}}$ improves over the ensemble performance of BERT$_{\text{medical}}$ by $\sim$ 1.5 points on ChemProt , $\sim$ 2 points on DDI, and $\sim$ 0.5 points on MedNLI and i2b2. However, we also see that the medical CharacterBERT model performs worse than its BERT analog in the specific case of ClinicalSTS, which we assume to be partly due to the presence of a few bad random seeds. Nevertheless, we can notice that the results with general-domain models seem to also be in favor of CharacterBERT as well. However, these differences may not be substantial after all. In an effort to provide a more rigorous comparison of these models, we perform statistical significance tests for each pair of models, namely, Almost Stochastic Order tests (ASO) (Dror et al., 2019), which we discuss in more detail in Section 3.3. Ultimately, given a pair of models A and B, the ASO test produces a value ϵ such that $\epsilon = 0$ when model A $\succeq$ B, $\epsilon = 1$ when B $\succeq$ A, and $\epsilon = 0.5$ when no order can be determined. Figure 5.10 shows the values of ϵ for all model pairs as well as for each task. Aside from these task-specific significance matrices, we also compute the average significance matrix as a way to provide a single aggregated signal[33]. Using this global matrix, we can see that CharacterBERT$_{\text{general}}$ improves over its BERT counterpart (row d, column c). Moreover, we also see that the overall best model is CharacterBERT$_{\text{medical}}$ as evidenced by the bottom blue row (cells $[f, a]$ to $[f, e]$). This is also shown on the single significance matrices with a blue bottom line for all tasks but, as expected, ClinicalSTS. All in all, the performed ASO tests provide additional evidence that CharacterBERT is indeed able to improve over vanilla BERT both as a general and specialized model.

Robustness to Noise and Misspellings

Since CharacterBERT uses multiple character-level CNNs to compute context-independent word representations, it may be expected that this model could be robust to different kinds of noise such as typos and misspellings. In order to investigate this idea, we create noisy versions of the MedNLI task where, given a noise level of X%, we transform each token from the original task corpus, with probability X%, into a misspelled version *via* either swapping two consecutive characters or by removing, adding or replacing a single character. We then use these corpora to run two sets of evaluations: one where the noise is added to the training, validation as well as the test set of the MedNLI task; and one where we only add noise to the test. We consider these two settings in an attempt to see, in case CharacterBERT proves to be more robust than BERT, whether this difference disappears when BERT is given the chance to adapt to the additional

[33]Note that the average significance score may not be seen as an actual significance score *per se*. However, this aggregated score may be useful for providing a global signal.

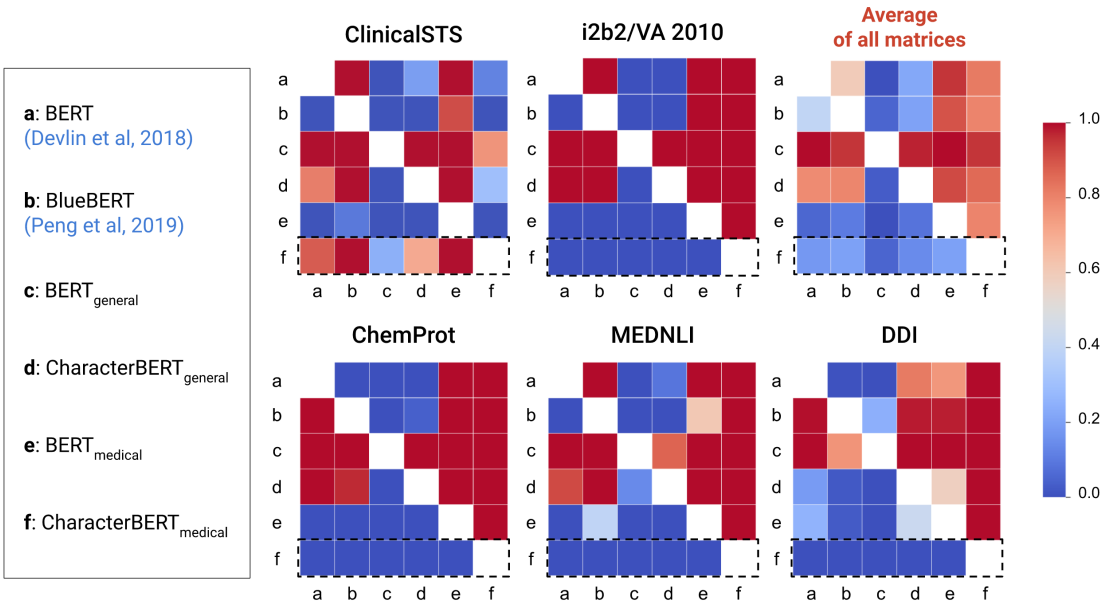

Figure 5.10: Statistical significance: Minimal distance ϵ for Almost Stochastic Order at level $\alpha = 5\%$. **Blue** cells mean that the left model is significantly better than the bottom model. **Red** cells mean the opposite.

noise.

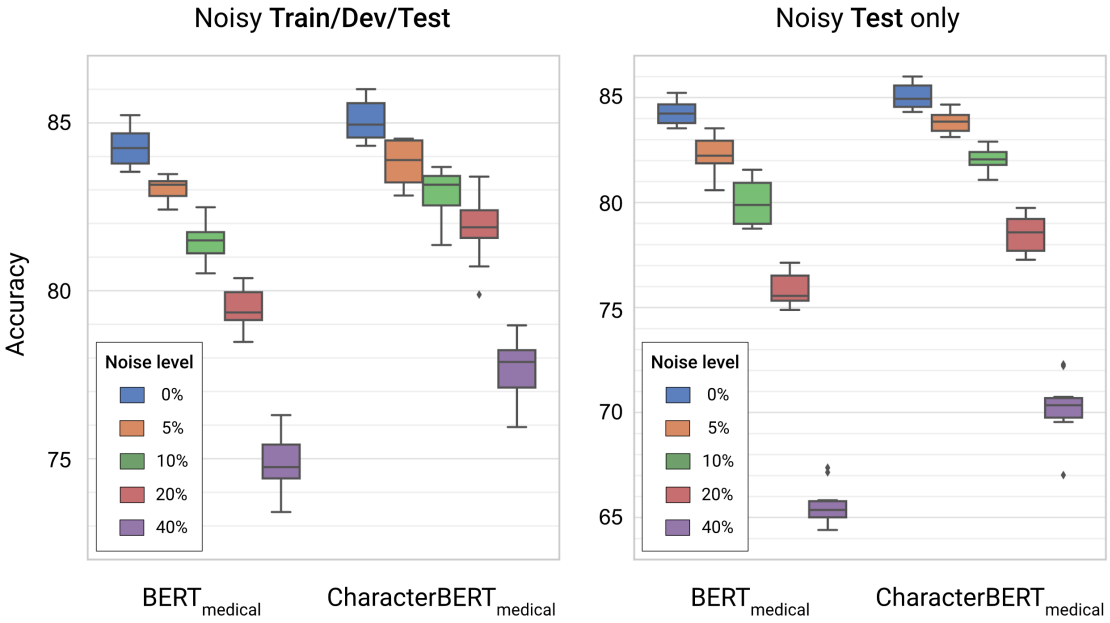

Figure 5.11: Comparing BERT and CharacterBERT on noisy (misspelled) versions of MedNLI test.

Figure 5.11 shows the results for $\text{BERT}_{\text{medical}}$ and $\text{CharacterBERT}_{\text{medical}}$ using various levels of noise on either the test set only or on all splits of the task dataset. As expected, we see that CharacterBERT is indeed more robust to misspellings, as evidenced by the slower decrease in performance compared to BERT in both situations. In particular, when a noise level of 40% is applied to the test set only, CharacterBERT is $\sim$ 5 F1 points higher than BERT whereas the original difference between the two models is only $<$ 1 F1 point. When both models are given the chance to adapt to the additional noise (i.e. noisy train/dev/test), the overall loss in performance when increasing the noise level is less important. However, the difference between BERT and CharacterBERT continues to increase along with the increase noise, confirming once again that CharacterBERT is more robust in comparison.

Summary

Overall, CharacterBERT seems to either perform at the same level or improve over the vanilla BERT architecture. This is especially true for the specialized versions and is further validated by ASO significance tests. The new variant is open-vocabulary and produces word-level representations. Moreover, it exhibits signs

of superior robustness to different forms of misspellings. This improved robustness is desirable as the original BERT seems to be sensitive to such misspellings (Pruthi et al., 2019; Sun et al., 2020). On the downside, CharacterBERT is slower to pre-train, although not as slow to fine-tune and even slightly faster at inference time. Future work may focus on applying a similar CharacterCNN to more recent transformer-based models such as ALBERT (Lan et al., 2020) or ERNIE (Zhang et al., 2019). Other directions may attempt to optimize the pre-training architecture to improve its speed or explore any other advantages that a character-level system may provide over WordPieces. It is however fair to note that other character-based Transformer models have been proposed around the same time as CharacterBERT. These are, for instance, CharBERT (Ma et al., 2020), which relies on both character and subword information; Charformer (Tay et al., 2021), which automatically learns latent subword representations using the character information; and CANINE (Clark et al., 2021), which can operate without any tokenization on raw character sequences.

5.4 Conclusion

BERT has definitely had a major impact on the way modern NLP is practiced, with most modern systems relying on either BERT or one of the several variants that has been spawning, attempting to improve over the original architecture in some specific way. In most cases, the tokenization procedure used by these models remains the same and relies on a pre-defined vocabulary of subwords to handle any new unseen tokens. While this system may be effective for dealing with the same general domain these WordPiece vocabularies are usually built for, issues may arise when re-training general models on in-domain corpora in an attempt to handle new specialized domains such as the medical domain. In fact, we have seen that a general vocabulary tends to produce seemingly less meaningful decompositions of domain-specific terms when compared to a specialized vocabulary, suggesting that a fully specialized model may be more suitable than a re-trained version relying on a general-domain model. While we verify this hypothesis to some extent, comparing a re-trained model using a general vocabulary to a medical model using a specialized vocabulary, it is clear, however, that building such specialized models from scratch is not as convenient as re-training pre-trained models from the general domain. In an attempt to take a step towards a BERT model that can be seamlessly re-trained on different domains without having to worry about a potential mismatch with the WordPiece vocabulary, we propose CharacterBERT, a variant that achieves an open-vocabulary and word-level model by consulting word characters. We show that in addition to being more convenient to use, this model overall outperforms a BERT analog that is trained in identical conditions, while

at the same time exhibiting signs of superior robustness to misspellings. However, CharacterBERT is slower to pre-train, although not as slow to fine-tune and slightly faster during inference. While it is still unclear whether this architecture is more suited for model transfer across different domains, we argue that it has, nevertheless, a few assets that weight in its favor: a unique byte vocabulary that can encode tokens from different languages as well as non-ascii symbols and a CharacterCNN that can be repeatedly re-trained to fit any new domain of interest.

Chapter 6

Specializing Representations Using In-domain Knowledge

6.1 Introduction

In previous chapters, we have touched upon the domain adaptation of word embeddings in the context of re-training general models on large specialized corpora. While this is probably the most popular and successful approach for constructing domain-specific word representations, we argue that knowledge bases—which are sometimes used to provide supervision (e.g. knowledge base completion tasks) or for evaluation purposes (e.g. in graph embedding evaluation)—may also be utilized as external sources of in-domain knowledge to enhance general-purpose word embeddings. In fact, there have been multiple efforts to leverage knowledge bases for improving the quality of word embeddings (see Background and Related Work, Section 2.3). However, these methods often focus on injecting linguistic knowledge and rarely explore how such external knowledge can be used to specialize pre-trained models to new domains. Moreover, existing methods either only apply to static representations or, instead, focus on modern language models and require complex procedures and/or architecture modifications. In this chapter, we consider external knowledge injection from a domain adaptation perspective. Specifically, we attempt to leverage in-domain knowledge bases to enrich existing representations, both traditional (e.g. static vectors) and modern (e.g. pre-trained language models), all the while focusing on keeping the overall method as simple as possible. In fact, we argue that by seeking a simple approach it is possible to develop strong baselines that can be refined easily by replacing any component from the global pipeline with improved and more advanced alternatives. In this spirit, investigate whether successful knowledge injection can be achieved using a simple approach comprising two components, namely: knowledge embeddings—and more

specifically, graph representation methods—which will be used to encode input knowledge bases into dense numerical representations; and meta-embeddings—specifically, vector concatenation—which will be used to combine existing word embeddings with pre-trained knowledge representations. Moreover, we explore how this simple approach can be generalized to deep neural language models by proposing a variant of BERT that incorporates external knowledge representations within its hidden layers, using a similar concatenation process. As for previous chapters, we focus on the medical domain in the specific context of the English language[1].

From a Transfer Learning perspective, this chapter explores yet another aspect of domain adaptation. In fact, while previous chapters aimed to investigate the impact of different hyperparameter choices on the quality of corpus-based domain-adaptive pre-training—namely, the corpus size vs. domain similarity tradeoff (§4) and, in case of BERT-like models, subword-tokenization (§5)—the current chapter explores how such domain adaptation can be achieved by leveraging external knowledge in the form of knowledge bases as well as hybrid meta-embeddings that combine both text and knowledge representations. Interestingly, these methods assume very little about the nature of the word representations and therefore, may also be used in conjunction with corpus-based techniques.

6.2 Knowledge Bases as Dense Representations

Let us assume that a knowledge base may be represented as a set of triples (h, r, t), where h and t are respectively the "head" and "tail" entities, and r is a (possibly directed) relation linking h to t. In this context, a fact such as "Paris is the capital of France" may appear in the knowledge base as (PARIS, `capital_of`, FRANCE). Given this definition, there are different ways to leverage the knowledge base information, which we broadly categorize into two groups: methods that use the triple information directly (e.g. retrofitting methods—§2.3.2), and those that use it indirectly by first converting the knowledge base into dense knowledge embeddings. In upcoming experiments, we investigate how existing word representations may be enhanced through the simple process of vector concatenation. Therefore, a necessary step is to first convert input knowledge bases into dense knowledge embeddings, which we develop in the next paragraph.

Knowledge Graph Embeddings There are several methods for embedding knowledge graphs (Ji et al., 2021). In their simplest form, these can rely on simple translational principles such as TransE (Bordes et al., 2013), which optimizes

[1]This work has led to an article that is currently under submission.

head, tail and relation embeddings according to the linear constraint $\hat{h} + \hat{r} \approx \hat{t}$; and TransR (Lin et al., 2015), which introduces separate spaces for entities $\{h, t\}$ and relations $\{r\}$ and uses a different constraint $M_r \cdot h + r \approx M_r \cdot t$, such that M_r is a relation-specific projection matrix which projects entities into the relation space. More recent methods, however, can be more complex and sometimes rely on neural networks like ConvE (Dettmers et al., 2018), which uses Convolutional Neural Networks; and KG-BERT (Yao et al., 2019), which relies on BERT in order to encode KB triples. In this chapter, we aim to provide a working yet simple solution for enhancing word representations using knowledge graphs. Consequently, we choose a simple method for converting input knowledge bases into dense representations, namely, one that only uses the relation information indirectly. In fact, most knowledge bases have a relation that is analog to `is_a`. This relation is usually the most frequent and provides useful information about parent-child relationships that connect the different entities of the knowledge base. As a first approximation, we focus on this single relation which allows to extract a a homogeneous graph of entities (i.e. as opposed to a set of KB triplets) which we leveraged using graph-level methods like NODE2VEC (Grover and Leskovec, 2016). These graph embedding methods encode the structure of the input graph into dense numerical vectors and may eventually be replaced with more advanced KB embedding models such as TransR and ConvE.

6.2.1 UMLS, MeSH & SNOMED CT

Given our focus on the medical domain, we consider the Unified Medical Language System (UMLS) (Bodenreider, 2004), a popular resource that involves multiple medical knowledge bases and ontologies. More specifically, the UMLS is a "Metathesaurus" that includes multiple subsets (a.k.a *vocabularies*), each organizing a specific set of medical concepts, as well as the different forms associated with these concepts (a.k.a *terms* or *strings*), according to a usually large number of varied relations (e.g. `active_ingredient_of`, `associated_with`, `branch_of`). Among the different vocabularies that are part of the UMLS, we select the Medical Subject Headings[2] (MeSH), which mainly organizes concepts and terms from the biomedical domain, as well as the Systematized Nomenclature Of Medicine - Clinical Terms[3] (SNOMED_CT), which is larger and has a focus on the clinical domain. Given these two vocabularies we can query[4] the UMLS and recover all pairs of Concept Unique Identifiers (CUI) for concepts and terms related through the `is_a` relation (e.g. `Chronic Bronchitis` is a `Chronic disease`). While there are many more types of relations that we can leverage, we focus on the single most

[2] https://www.nlm.nih.gov/mesh/meshhome.html
[3] https://www.nlm.nih.gov/healthit/snomedct/index.html
[4] SQL scripts are provided in the code repository.

frequent[5] type `is_a` in order to extract a single graph from each vocabulary and use the graph-level embedding method NODE2VEC.

6.2.2 Dense Representations with NODE2VEC

The NODE2VEC Method Roughly speaking, NODE2VEC (Grover and Leskovec, 2016) can be seen as an extension of WORD2VEC (Mikolov et al., 2013) for learning node representations. By using random walks on the input graph, this method generates a set of node sequences that can be used as pseudo word sequences, producing node embeddings *via* a WORD2VEC-like objective. While this approach is not specific to NODE2VEC *per se* and has in fact been explored in earlier methods like DeepWalk (Perozzi et al., 2014), the former introduces a more flexible type of random walks that allows it to explore a more diverse set of node neighbourhoods.

Learning NODE2VEC Embeddings for the UMLS We build a graph of concepts from each vocabulary by limiting ourselves to triples with the `is_a` relation and replacing head and tail entities with their Concept Unique Identifier (CUI). We then run the official `Python` implementation of NODE2VEC[6] with default parameters on each vocabulary graph, which produces 256-dimensional node (i.e. concept) representations. In total, this step yields 29,738 CUI embeddings for MeSH concepts and 389,872 CUI embeddings for SNOMED with 15,418 overlapping CUIs having both a MeSH and a SNOMED representation. The visualization of these embeddings using a PCA (see Figure 6.1) shows that this method is able to separate different categories of medical concepts in different subspaces, which suggests some level of encoded medical knowledge in the learned NODE2VEC representations.

Using NODE2VEC Embeddings in Practice While the NODE2VEC embeddings are interesting *per se*, we ultimately would like to combine these representations with existing word vectors in order to investigate whether this can benefit the performance on downstream tasks. As a result, we need to be able to map any input token with an appropriate knowledge representation using the learned NODE2VEC vectors. We adopt the following procedure:

1. Given our global vocabulary of concepts (CUIs), we concatenate both sets of knowledge embeddings (i.e. MeSH and SNOMED) in order to get a final 512-dimensional knowledge vector for each CUI. When a concept occurs only

[5] In the entire 2021AA version of the UMLS, there are $\sim$ 3.5 million pairs with the `is_a` relation. For comparison, the frequencies for the other top-10 most frequent types relations range from $\sim$ 1.7 million to $\sim$ 500K.

[6] https://github.com/aditya-grover/node2vec

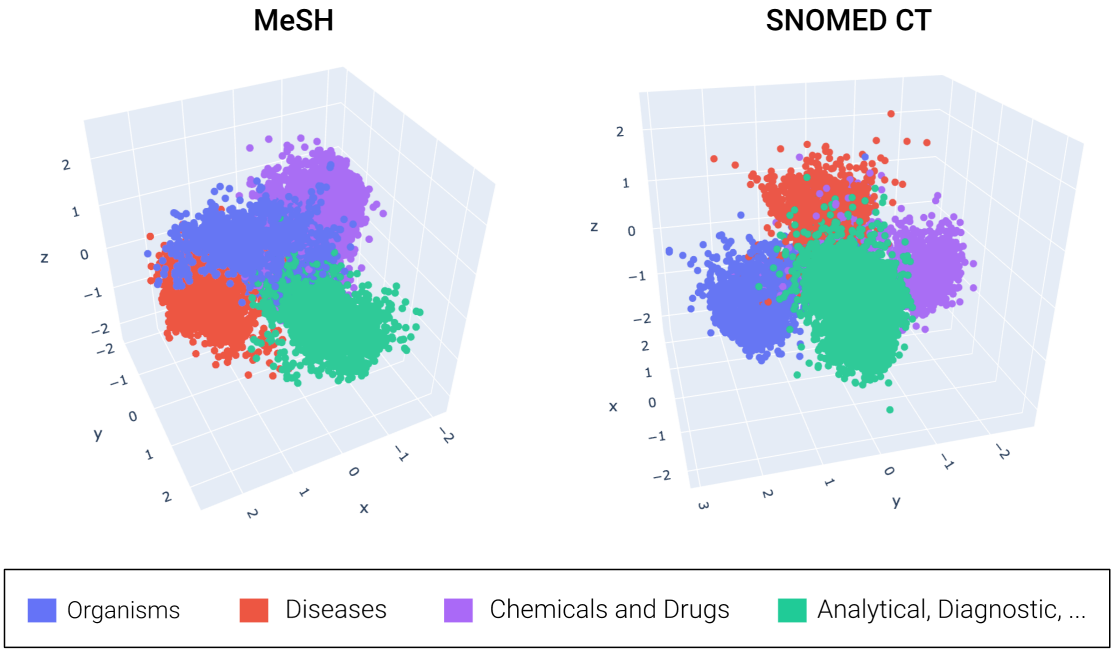

Figure 6.1: PCA of MeSH and SNOMED graph embeddings highlighting four different categories of medical concepts.

in either MeSH or SNOMED, we use zero-padding in place of the missing representation.

2. Using these representations requires locating concept mentions in the text, which refers to the task of concept normalization (a.k.a entity linking) whose aim is to identify the various linguistic forms that a given concept can take. In practice, concept normalization is a complex task, involving picking the right concept to disambiguate an entity mention among possibly several hundreds of thousands candidates from the knowledge base, all the while taking into account the different variations (i.e. strings) that may exist for each concept in the gold resource as well as the different forms that occur in natural text though casing, misspellings, etc. There are different systems for concept normalization which appear, for instance, in the various challenges that are organized on this topic[7]. However, in this work, we opt for a simple way to perform this normalization using an exact string matching between the reference linguistic forms from the UMLS[8] (i.e. strings/terms) and the target texts. This usually allows to disambiguate a reasonable amount of mentions, and the use of stronger or more advanced entity linkers is left for future work.

[7]See for instance: https://n2c2.dbmi.hms.harvard.edu/track3
[8]Which are available in the MRCONSO table of the metathesaurus.

3. Once our concept vocabulary has been projected onto the input texts, we associate each token with its concept representation, using zero-valued vectors for any tokens that are "out-of-entity".

6.3 Simple Approach to Knowledge Injection

Now that we have a way to compute concept representations and to associate them to our input text, we return to our investigation on whether existing word representations can be improved using external knowledge and a basic set of tools, namely: graph-level knowledge embeddings and concatenation. To find out whether this is possible, we consider different static and contextual (i.e. neural language models) representations from both the general and medical domains, and conduct several evaluations on a wide range of clinical and biomedical tasks.

6.3.1 Embedding Specialization Methods

Static Representations

Although static representations have been arguably outclassed by more recent contextual language models, we argue that these methods are still useful in situations where large neural models cannot be implemented due to the lack of resources, as well as in the context of intrinsic tasks such as word and sentence similarity. Moreover, including these static representations in our analysis allows us to see whether our approach behaves differently on this simpler class of models before delving into the generally more complex neural representations. In practice, we use the fastText method to learn word representations on corpora from different domains, then attempt to specialize these embeddings through concatenation, combining the word and knowledge vectors at the token-level. Since we have already trained fastText representations in Section 4.2, we can simply re-use these representations, for which we recall the different source corpora:

Gigaword (Graff et al., 2003) A newswire corpus including sources such as the New York Times. It is a general-domain corpus with $\sim$ 1 billion tokens.

PubMed (MEDLINE) A corpus of biomedical article abstracts. This is a medical (biomedical) domain corpus with $\sim$ 2 billion tokens.

MIMIC (Johnson et al., 2016) A corpus of clinical notes from several hospitals. This is a medical (clinical) domain corpus with $\sim$ 0.5 billion tokens.

Contextual Representations

We also experiment with contextual embeddings since these are the most popular option today. Ultimately, we would like to be able to improve over state-of-the-art neural language models which may or may not rely on a TRANSFORMER architecture. In our experiments, however, we consider BERT (Devlin et al., 2019) and CHARACTERBERT (El Boukkouri et al., 2020)[9]. The former is included as a strong baseline for modern transformer-based embeddings and the latter is included as a word-level variant that performs well in the medical domain (see Section 5.3). Furthermore, considering two different models allows us to have a larger sample size for measuring the impact of our strategies on transformer-based representations. In practice, we specialize these models in two different ways:

Concatenation Similar to static representations, we combine WordPiece representations coming from BERT or token representations coming from CHARACTERBERT with the appropriate knowledge vector as explained in Section 6.2.2.

Knowledge Injection Modules (KIM) We propose these small neural layers in an attempt to generalize the idea of concatenating word and knowledge embeddings to the internal states of a transformer-based model. When placed after any given layer, this module concatenates the hidden representations from that layer h_i with the appropriate knowledge representations KG_i. Then, it projects this concatenation to recover a set of new states which we call "enhanced states" $\mathbb{h}_i$, with the same dimensionality as the original hidden representations. Since this operation could lead to some loss of the information originally available in the hidden states, we compute a linear combination of the enhanced and original states using trainable parameters $\alpha \in [0, 1]$ and $\beta = 1 - \alpha$. Finally, we feed the resulting representation $\boldsymbol{h}_i$ to the next model layer. In summary:

$$\boldsymbol{h}_i = \alpha\ \mathbb{h}_i + \beta\ \mathrm{h}_i$$

where $\mathbb{h}_i = [\mathrm{h}_i; \mathrm{KB}_i]\ W + b$ and W, b are respectively the weight matrix and bias of the linear projection operation (see Figure 6.2). Our KIM layers are loosely related to the idea of Adapter Modules (Houlsby et al., 2019; Wang et al., 2021) which we discuss in Section 2.2.2. However, our modules are conceptually simpler and built specifically to incorporate external representations into the hidden states of transformer-based models.

[9] We use the "base-uncased" versions of these models.

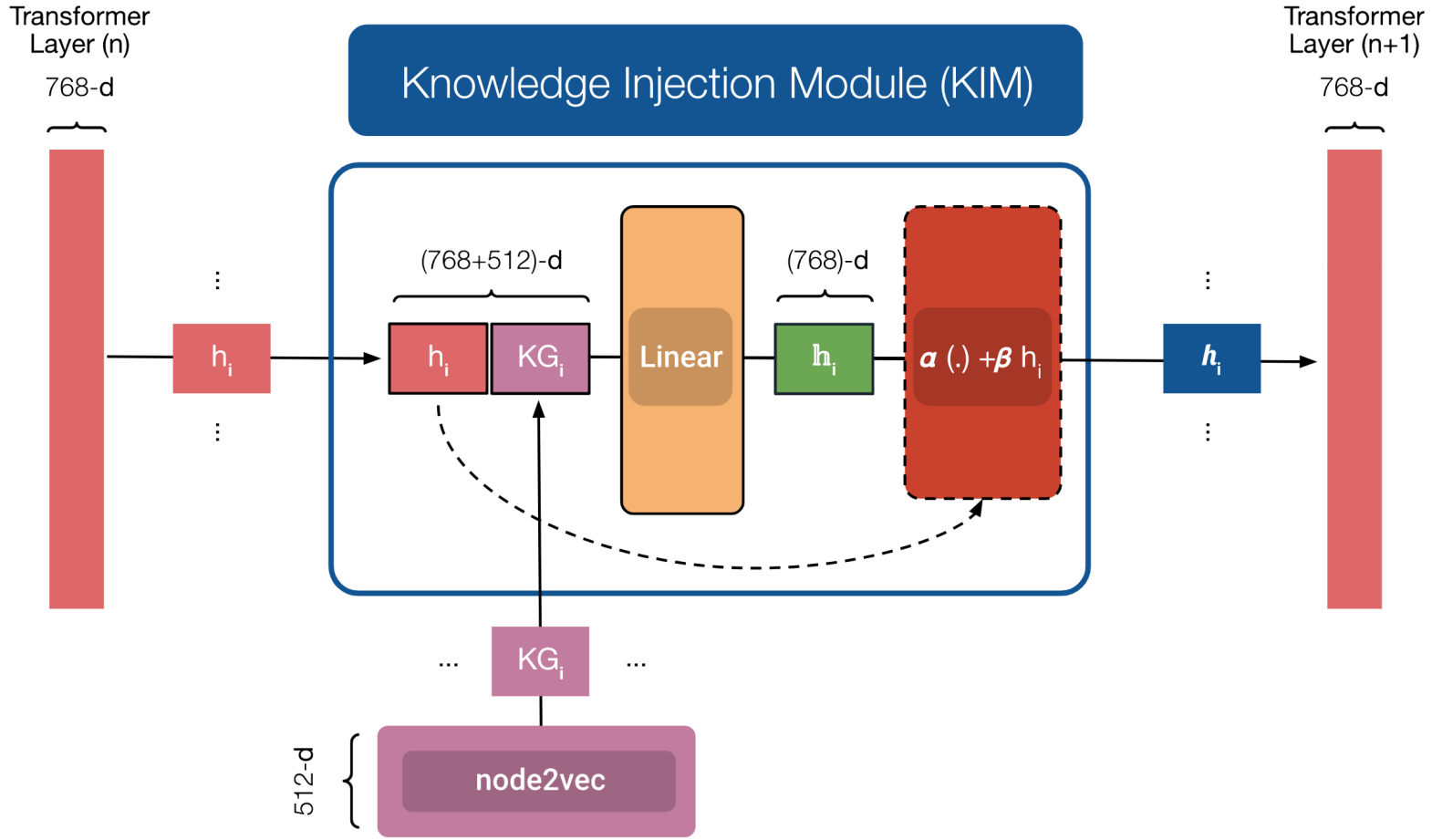

Figure 6.2: Detailed view of a Knowledge Injection Module (KIM) between two TRANSFORMER layers. Given an incoming hidden (h_i) and knowledge representation (KG_i), the module concatenates both vectors ($[\mathrm{h}_i; \mathrm{KG}_i]$), applies a linear projection down to the original size ($\mathbb{h}_i$), then computes a mixture of the enhanced and original states using parameters $\alpha \in [0, 1]$ and $\beta = 1 - \alpha$. The output ($\boldsymbol{h}_i$) is ultimately fed to the next TRANSFORMER layer.

6.3.2 Experiments

Embedding Configurations

In what follows, and unless explicitly stated, *Model* may be either a BERT, CharacterBERT or fastText model. With that in mind, these are the various embedding configurations that we evaluate:

Random These are randomly initialized 256-dimensional static embeddings used as a baseline for static word representations. Moreover, when combined with node2vec vectors, these representations are supposed to provide us with insights on the raw contribution of external knowledge embeddings.

Model These can be either 256-dimensional static embeddings of the form fastText(*corpus*), where *corpus* is one of the corpora presented in Section 6.3.1, or 768-dimensional BERT or CharacterBERT representations.

[*Model*, node2vec] This is the token-level concatenation of *Model* with the pre-trained 512-dimensional node2vec representations from Section 6.2.2.

***Model*(medical)** When *Model* is either BERT or CharacterBERT, this is a medical model resulting from the re-training of a general version of *Model* on a large medical corpus consisting of $\sim$ 0.5 billion tokens from MIMIC-III and $\sim$ 0.5 billion tokens from PMC-OA biomedical abstracts.[10]

Enhanced*Model*(medical) When *Model* is either BERT or CharacterBERT, this is the same configuration as *Model*(medical) but this time the architecture is modified to use a randomly initialized Knowledge Injection Module (KIM) after every Transformer layer, as well as either the WordPiece embedding layer for BERT, or the CharacterCNN for CharacterBERT.[11]

For the last two configurations, we follow a standard pre-training procedure that comprises both Masked Language Modeling (MLM) and Next Sentence Prediction (NSP), and adapt the implementation from Section 5.3 while keeping similar hyperparameters.[12]

Evaluation Tasks

Insights from model evaluation can be misleading, especially when only a limited number of tasks is considered. In an effort to conduct a more thorough evaluation

[10] This is the same corpus that we have used in previous experiments (see Section 5.3).

[11] More precisely, we add the Knowledge Injection Module after the static token representations have been combined with the segment and position embeddings.

[12] A `bash` script including the exact hyperparameter values is available at this URL.

	Entity Recognition			Rel. Extraction		Semantic Similarity		NLI
	i2b2	BC5-Dis.	BC5-Chem.	ChemProt	DDI	BIOSSES	ClinicalSTS	MedNLI
Train.	22.263	4.182	5.203	4.154	2.937	64	600	11.232
Val.	5.565	4.244	5.347	2.416	1.004	16	150	1.395
Test	45.009	4.424	5.385	3.458	979	20	318	1.422

Table 6.1: Number of entities, positive relations or samples for each task.

of our models, we consider various tasks from both the biomedical and clinical domains, ranging from entity recognition to sentence similarity. More specifically, we consider previously used evaluation tasks (see Section 5.2.2) as well as two additional sequence labeling tasks and a sentence similarity task from the biomedical domain (see Table 6.1).

i2b2 We reuse the i2b2/VA 2010 clinical concept extraction task (Uzuner et al., 2011), which aims to detect three categories of clinical entities: PROBLEM (e.g. "headache"), TREATMENT (e.g. "oxycodone") and TEST (e.g. "MRI"). This uses Exact match F1 measure as an evaluation metric.

BC5-Disease/Chemical These are two additional sequence labeling tasks from BioCreative V CDR (Li et al., 2016), which respectively aim to detect DISEASE (e.g. "hepatitis") and CHEMICAL (e.g. "corticosteroid") entities. It uses Exact F1 as a metric.

DDI We reuse this relation extraction task, which we recall focuses on classifying drug-drug interactions into five categories: ADVISE (DDI-advise), EFFECT (DDI-effect), MECHANISM (DDI-mechanism), INTERACTION (DDI-int), and DDI-false for no interaction. It uses the micro-averaged F1 over non-negative classes as a metric.

ChemProt We also reuse this relation extraction task, which focuses on classifying chemical-protein relations into six categories: ACTIVATOR (CPR:3), INHIBITOR (CPR:4), AGONIST (CPR:5), ANTAGONIST (CPR:6), SUBSTRATE (CPR:9) and FALSE for no relation. The micro-averaged F1 measure over non-negative classes as metric.

BIOSSES This is an additional and small (i.e. 100 samples total) sentence similarity dataset in the biomedical domain (Soğancıoğlu et al., 2017). This uses the Pearson correlation of predicted and gold similarities is used as a metric.

ClinicalSTS We reuse this clinical domain sentence similarity task. Naturally, this uses Pearson correlation as well.

MEDNLI We reuse this clinical natural language inference task, which aims to classify pairs of sentences into three categories: ENTAILMENT, CONTRADICTION, and NEUTRAL. Here we use classification accuracy as a metric.

Evaluation Architectures

Since we will be fine-tuning models that rely on either static or BERT-like representations, it is important to clearly state which architectures are used in which situations since these can differ greatly depending on the kind of embedding as well as the nature of the evaluation task (e.g. sentence similarity vs. sequence labeling). Moreover, being aware of these different architectures may also be useful for interpreting potential differences in behaviour in these different settings.

Sequence Labeling For tagging tasks, we use a classic architecture comprising an encoder followed by a classification layer and a Conditional Random Field layer (CRF) (Lafferty et al., 2001). Depending on the situation, the encoder is changed to adapt to the type of word representations: **FASTTEXT** are fed to a Bi-LSTM[13]; **[FASTTEXT, NODE2VEC]** are concatenated prior to being fed to a Bi-LSTM as well; variants of **BERT** and **ENHANCEDBERT** are considered as their own encoders and are used as-is. Finally, variants of **[BERT, NODE2VEC]** concatenate token and knowledge representations before feeding them forward. In this last setting, it is important to note that the transformer representations are contextualized prior to being combined with the NODE2VEC vectors while all other configurations involving knowledge representations (e.g. **[FASTTEXT, NODE2VEC]**, **ENHANCEDBERT**) are combined with NODE2VEC before the contextualization step, which happens either through a Bi-LSTM or *via* TRANSFORMER layers.

Classification The architecture for relation extraction tasks, which we frame here as classification tasks, is similar but requires a summarized representation at the example-level. As a result, the encoder needs to aggregate token vectors to produce a single summarized representation for each input text. Here again, **FASTTEXT** and **[FASTTEXT, NODE2VEC]** are fed to a Bi-LSTM, but this time, the output is average-pooled to produce a single feature vector. With variants of **BERT** and **ENHANCEDBERT**, we follow the common practice and use the pooler representation[14]. Finally, when

[13] In all future mentions of a Bi-LSTM, we will be referring to a 3-layer network with 50% recurrent dropout and an overall output size of 512.

[14] We recall that BERT expects an input that starts with a special symbol called [CLS]. This symbol's representation is supposed to encode global information about the model's input, however, in practice, the [CLS] vector goes through an additional dense layer called the pooler before being used as a feature vector for sentence-level downstream tasks.

using variants of **[BERT, NODE2VEC]**, the knowledge representations are combined through average-pooling and the resulting vector is concatenated with BERT's aforementioned pooler representation.

Natural Language Inference For NLI tasks, we require a global summarized representation at the sentence-pair level that we can ultimately feed to a classification layer. For **static embeddings**, we compute an average-pooled Bi-LSTM representation for the first sentence u as well as for the second one v, then compute a global feature vector $[u, v, |u - v|, u * v]$ following the approach of InferSent (Conneau et al., 2017b). When using variants of **BERT** and **ENHANCEDBERT**, we simply use the pooler representation as these models can accept sentence pairs. Finally, with variants of **[BERT, NODE2VEC]**, we concatenate the pooler output with InferSent-style features computed using the NODE2VEC knowledge representations.

Sentence Similarity For STS tasks, the overall approach is different depending on whether we use static or contextual representations. For **static embeddings**, we follow the common practice and compute a bag-of-word representation for each sentence; then, we measure the cosine similarity between the two representations. Perhaps unusually, we frame these models in a supervised setting and backpropagate errors to adapt the static representations during training. When **contextual embeddings** are involved, we frame the sentence similarity tasks as regression problems and use the same encoder as for classification, adding the appropriate output layer to produce similarity scores. In the specific case of **[BERT, NODE2VEC]** variants, just like for STS, we concatenate the pooler output with NODE2VEC InferSent-style features.

We provide illustrations for each of these architectures in the Appendix B.

Evaluation Setup

We fine-tune all model parameters, including the static and NODE2VEC vectors, using the following hyper-parameters:

- *Validation Set*: when no validation set is available, we use 20% of the training data as validation data.
- *Epochs*: we train each model for 15 epochs for all tasks, except BIOSSES and CLINICALSTS for which we run 100 and 50 epochs respectively[15].

[15]This also means that the CLINICALSTS results are expected to differ from those of previous chapters where a different number of epochs (i.e. 15) was used.

- *Batch Size*: we use batches of 32 examples.
- *Optimizer & Learning Rate*: we use the Adam optimizer (Kingma and Ba, 2015) with a learning rate of 1e-3 for all **non-transformer weights** and a learning rate of 3e-5 for **transformer weights**, for which we also use a weight decay of 0.1 and a linear schedule with 10% warm-up.

As for all previous experiments, we account for some of the randomness during the fine-tuning procedure by evaluating each model, on each task, using 10 different random seeds. We then use these seeds as the basis for computing ensemble scores as well as statistical significance tests (see Chapter 3.3 for more details).

6.3.3 Results & Discussion

	i2b2	bc5-disease	bc5-chemical	chemprot	ddi	biosses	clinical_sts	mednli
Random	83.42	74.12	75.91	51.72	64.92	**64.42**	62.09	**70.68**
[Random, node2vec]	**84.57**	**80.93**	**84.91**	**53.66**	**65.14**	59.83	**63.72**	70.36
fastText(Gigaword)	84.70	76.79	82.76	52.06	62.83	**82.43**	**71.93**	69.66
[fastText(Gigaword), node2vec]	**84.99**	**80.86**	**86.50**	**52.64**	**64.44**	53.55	65.95	**70.08**
fastText(PubMed)	85.16	79.71	88.67	**54.62**	66.17	**91.49**	**72.10**	70.51
[fastText(PubMed), node2vec]	**85.49**	**82.62**	**89.45**	54.36	**67.11**	62.32	67.46	**70.82**
fastText(MIMIC)	85.49	78.92	84.93	51.54	67.12	**76.94**	**72.42**	**71.74**
[fastText(MIMIC), node2vec]	**86.01**	**80.70**	**86.38**	**52.90**	**67.59**	51.72	69.17	70.96
BERT	**88.16**	79.56	88.63	**71.75**	**79.95**	81.94	**84.71**	**79.75**
[BERT, node2vec]	87.76	**80.66**	**88.88**	71.36	79.60	**85.93**	84.34	79.30
BERT(medical)	**89.45**	81.88	**90.67**	**71.96**	**79.79**	89.14	**84.21**	**83.66**
EnhancedBERT(medical)	89.40	**83.48**	90.36	71.22	78.74	**92.12**	83.59	83.03
CharacterBERT	**88.08**	80.90	88.73	70.61	79.42	90.58	84.49	78.85
[CharacterBERT, node2vec]	87.81	**81.63**	**89.39**	**71.01**	**81.23**	**91.03**	**84.89**	**79.19**
CharacterBERT(medical)	**89.82**	83.60	92.07	**73.63**	**80.67**	87.52	83.63	**84.66**
EnhancedCharacterBERT(medical)	89.76	**85.05**	**92.08**	73.01	79.39	**92.65**	**84.42**	84.46

Table 6.2: Performance of model ensembles on evaluation tasks from the medical domain. Results are displayed in pairs: baseline model on the top line and specialized version (either through concatenation or KIM) on the bottom line. The colors show statistical significance, with bluer colors meaning the specialized models improve more significantly over the baselines and redder colors showing a more significant degradation in performance.

For better visibility, and given the large number of experiments, we organize our results in pairs composed of a baseline along with our specialized version of that baseline. We report the average ensemble performances of each model pair as two consecutive rows with the baseline on top (see Table 6.2). We also emphasize in bold the best performance, within each pair, on each task (column). Moreover, we color the specialized version according to its ASO distance (ϵ) to the baseline model. These colors range from red ($\epsilon = 0$) for a significant degradation, to blue ($\epsilon = 1$) for a significant improvement. Please note that the font weight and color system show related yet different kinds of information (i.e. scores vs. statistical significance) and that the color coding for these results is opposite to what we have used in previous sections (i.e. contrary to §5.3.3, here we use: blue > red).

Random vs. [Random, NODE2VEC] It is interesting to note that randomly initialized static embeddings manage to achieve reasonable results, sometimes even outperforming pre-trained FASTTEXT representations (see Random vs. {GIGAWORD or PUBMED} on MEDNLI). However, given the random nature of these vectors, we can expect in-domain knowledge representations to be able to easily improve the baseline performance on downstream in-domain tasks. While this is verified in most situations (see I2B2 through CHEMPROT), we also note a degradation on BIOSSES and MEDNLI. While it is difficult to propose a rigorous explanation for this phenomenon, it is nonetheless interesting as it could point to situations where the external knowledge may not be relevant to the task at hand (notice, for instance, the majority of red cells in the MEDNLI column).

FASTTEXT(X) vs. [FASTTEXT(X), NODE2VEC] Overall, using concatenation to combine NODE2VEC knowledge representations with FASTTEXT embeddings seems to result in consistent gains, notably on tagging and classification tasks (see top-left section of the table). Moreover, these results seem to hold regardless of the domain of origin, as word embeddings trained on GIGAWORD (general domain), PUBMED (biomedical domain) and MIMIC (clinical domain) all seem to benefit from this combination. However, we can also see that the results on STS are significantly worse, with drops of up to 30 points of correlation on BIOSSES with FASTTEXT(GIGAWORD). This degradation may be explained by the fact that the "bag-of-word + cosine similarity" rests on the idea that sentence representations can be constructed by summing the vectors of individual words within a sentence. While this is a usual practice when dealing with distributional representations like WORD2VEC, it is however unclear that this is suited for knowledge base representations computed using NODE2VEC. Therefore, our STS architecture may not be adapted to meta-embeddings made of both word and knowledge representations and could explain our overall failure to produce satisfactory sentence similarity

models. Moreover, we note that the NODE2VEC vectors are rather sparse (most concepts do not have both a MeSH and SNOMED representation) and twice as large as the word representations, which may further contribute to the overall negative effect as, for instance, a majority of zeros can easily distort the average representations computed in the context of bag-of-word systems.

BERT vs. [BERT, NODE2VEC] Looking at the results for general-domain contextual embeddings (i.e. BERT, [BERT, NODE2VEC] and their CHARACTERBERT analogs), we notice multiple instances where the concatenation with NODE2VEC proves to be beneficial (see BC5 and BIOSSES for BERT and all tasks but I2B2 and MEDNLI for CHARACTERBERT). However, there also seems to be some discrepancies where this concatenation sometimes improves the CHARACTERBERT baseline on one hand, but impairs the BERT baseline on the other (see CHEMPROT and DDI). A closer look at these situations shows that plain CHARACTERBERT performs slightly lower than plain BERT[16], which could explain why CHARACTERBERT is improved by the NODE2VEC concatenation while BERT is not. In fact, if we suppose that the baseline CHARACTERBERT model may be missing some information that is relevant for these specific tasks[17], then we may assume that the knowledge representations compensate for this missing information which, supposedly, may be already available in the baseline BERT model.

BERT(medical) vs. ENHANCEDBERT(medical) Adding Knowledge Injection Modules to the original architectures of BERT and CHARACTERBERT seems to produce different results depending on the evaluation task. In fact, we can see that ENHANCEDBERT and ENHANCEDCHARACTERBERT respectively lose 1.05 and 1.28 F1 relative to their baselines on the DDI task. However, we also notice that these same models lead to gains of 1.6 and 1.45 F1 on the BC5-DISEASE task. Incidentally, the BC5 tasks are particularly interesting as they are based on the exact same corpus but focus on two different types of entities: DISEASE and CHEMICAL. As a result, given that ENHANCEDBERT(medical) performs better than BERT(medical) on BC5-DISEASE and worse on BC5-CHEMICAL, it is safe to assume that this is not due to the Knowledge Injection Modules being inherently harmful but rather to the information available in the knowledge representations themselves being, relative to what is already available in the base model, more relevant for the first task than for the second one. Consequently, we may assume

[16]Note that this BERT model is the original version from (Devlin et al., 2019) and not the one we train ourselves in Section 5.2.2

[17]Recall that our general-domain BERT performed below original BERT, which we assumed to be due to differences in training and corpora. For similar reasons, general CHARACTERBERT may be missing some knowledge that is available in original BERT.

that the Knowledge Injection Modules can be useful for incorporating external information into a model but that the final downstream performance may depend on how relevant this information is for any given task. Moreover, it is important to note that the baseline models for EnhancedBERT and EnhancedCharacterBERT are already specialized. In fact, we recall that BERT(medical) and CharacterBERT(medical) are the result of re-training general models on large in-domain corpora, which may explain why it is seemingly more difficult to achieve further specialization using external knowledge with these models.

6.3.4 Summary

Overall, our approach seems to be sufficient for improving existing representations using external knowledge. In fact, we show that encoding knowledge bases via node2vec then using concatenation to enhance existing word embeddings leads to consistent results for static representations, with the exception of sentence similarity tasks for which we argue that the "bag-of-words + cosine similarity" is not fully appropriate. Moreover, we also demonstrate that the plain concatenation can be successful for specializing transformer-based models. However, we suggest that the injected knowledge be chosen appropriately so that it is relevant enough for the target task. Finally, we show multiple instances where the proposed Knowledge Injection Modules successfully improve over a vanilla medical-domain model. However, we insist here again on the relevance of the injected knowledge.

6.4 Conclusion

In this section, we focused on exploring the extent to which specialized information from a knowledge graph could be injected into existing word embeddings using a very simple set of tools: graph embeddings and concatenation. While focusing on the medical domain in the English language, we conducted multiple evaluations on tasks ranging from entity recognition to sentence similarity. These evaluations demonstrated that the concatenation with in-domain graph representations can be a simple yet effective approach to model specialization, with significant gains on multiple settings. Moreover, applying the same process of concatenation within transformer-based contextual models proved to be beneficial as well, with notable improvements using Knowledge Injection Modules (KIM) on several downstream tasks. Notwithstanding, our goal was to construct a strong enough baseline using simple components so that straightforward improvements can be achieved using more elaborate methods. As a result, there is still room for improvement.

Knowledge Embeddings As mentioned in Section 6.2.1, there are many more types of relations that we could use to improve the quality of the knowledge representations. An interesting path may be to use recent meta-embedding methods like Word Prisms (He et al., 2020a), in conjunction with knowledge embedding methods that learn different concept representations for different kinds of relations like TransR (Lin et al., 2015), to learn multi-faceted knowledge representations.

Pre-training Objective Monitoring the values of the linear combination weights within enhanced versions of BERT and CharacterBERT (recall α, β from Figure 6.2 and see Figure 6.3), we notice that the final contribution α of the "enhanced states" after pre-training is noticeably small ($< 5\%$). This may indicate that the external knowledge was not very useful during pre-training, probably leading the final models to probably under-utilize this information. Future work may define additional objectives to encourage the model to exploit these external representations more strongly.

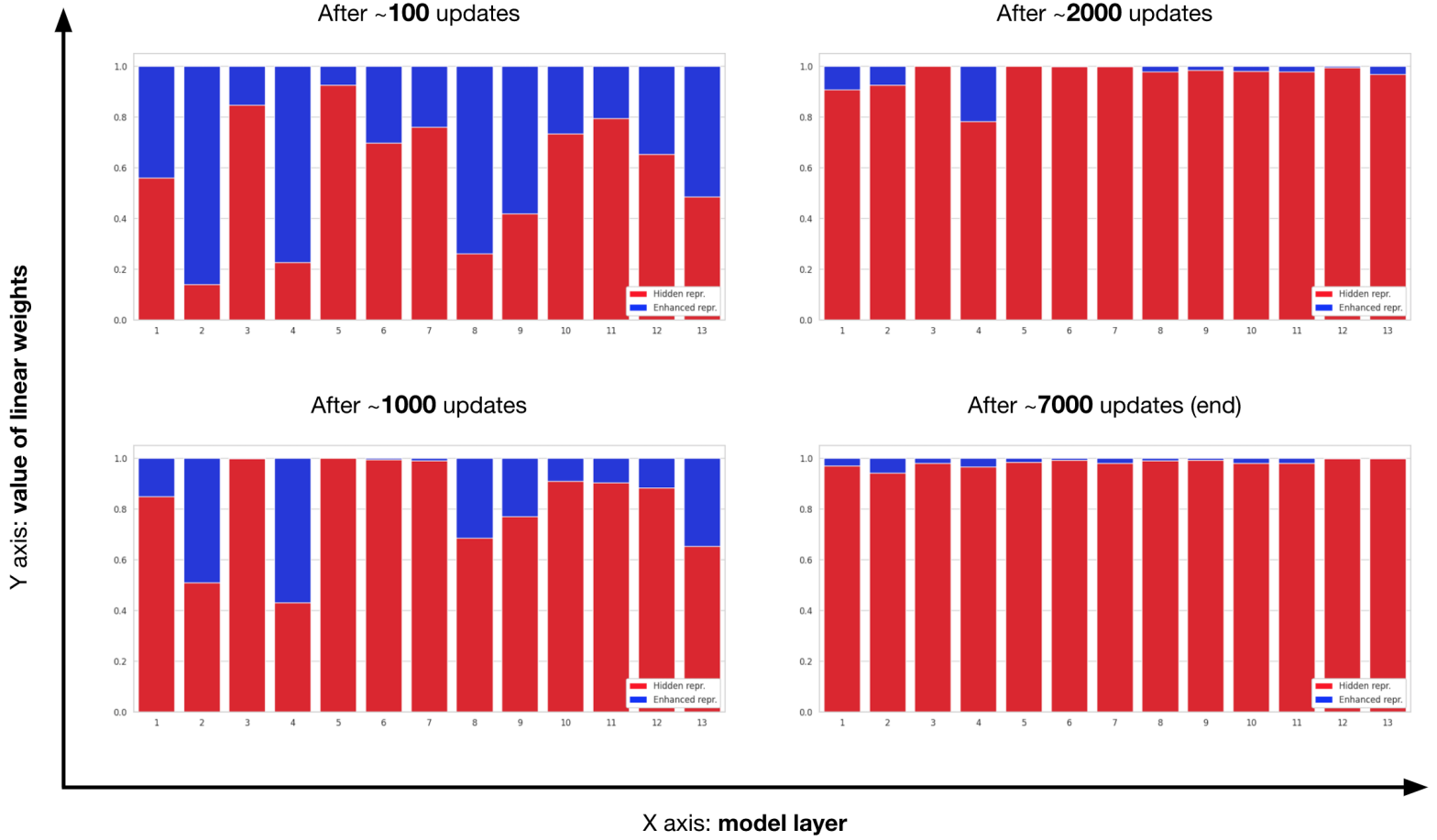

Figure 6.3: Evolution of the linear combination weights inside each Knowledge Injection Module when training the EnhancedCharacterBERT architecture on a medical corpus. The contribution of the enhanced states α is shown in **blue** and the contribution of the original hidden states β is shown in **red**. Each figure is a barplot showing these contributions α, β for the context-independent embeddings' KIM ($x = 1$) as well as for all subsequent transformer layers' KIMs ($x \in \{2, 3, \ldots, 13\}$)

Chapter 7

Conclusion

This chapter provides an overview of the different contributions we have made throughout this thesis as well as possible future directions which we think may be interesting to explore.

7.1 Summary

In this dissertation, we have investigated different ways to improve general out-of-the-box word representations using in-domain corpora and knowledge bases. In Chapter 2, we have provided relevant background knowledge on Transfer Learning and Domain Adaptation as ways to leverage related tasks and corpora. Moreover, we have touched upon methods for leveraging external knowledge like retro-fitting and introduced meta-embeddings which we ultimately use to enhance word representations with in-domain knowledge.

Subsequently, in Chapter 3, we provided an overview of word embedding evaluation, discussing intrinsic and extrinsic approaches then sharing our perspective on the matter. We also reviewed the different ways in which we attempt throughout this thesis to perform fair and rigorous evaluations, mentioning random restarts, model ensembles and statistical significance.

In Chapter 4, we focused on how text corpora can be used to improve existing embeddings and confirmed that both corpus size and domain similarity play an important role in this process. We then proposed a method for leveraging a small in-domain corpus along with other widely available resources to produce results that are on par with training in-domain representations.

Chapter 5 goes on a slight tangent and tackles the topic of BERT-like models and the WordPiece vocabulary. In this chapter, we demonstrated that the original BERT vocabulary may not be suited for specialized domains then trained parallel models from scratch, on corpora from different domains, using different

vocabularies. This led us to conclude that the slight improvements made by the fully specialized models were not worth the trouble of training such models from scratch and suggest in practice to re-train general versions. Moreover, we proposed a new variant of BERT that does not rely on WordPieces, using instead a character module inspired from ELMo. We then showed that this new variant, called CharacterBERT, outperformed a BERT model trained in similar conditions while at the same time being more robust to misspellings.

Finally, in Chapter 6 we returned to the topic of enhancing word representations with in-domain information, this time focusing on using knowledge bases. With simplicity in mind, we developed a strong baseline consisting in knowledge base embeddings and concatenation. We showed that this method can lead to significant gains when applied to static word representations as well as transformer-based models like BERT and CharacterBERT. Moreover, we proposed Knowledge Injection Modules (KIM) to directly incorporate knowledge representations within the architecture of BERT-like models. We showed that these modules may lead to some improvements as well, however, we argue that such improvements are probably largely dependent on the relevance of the chosen external knowledge to the target task.

7.2 Contributions

Throughout this thesis we have made the following contributions:

Corpus Size vs. Similarity In Section 4.2, we confirm the result that both corpus size and similarity to the target task domain have a positive impact on the downstream performance. Moreover, we also show that given sufficiently large corpora, increased domain similarity seems to be more desirable than increased size. However, what constitutes a sufficiently large corpus is still unclear.

Leveraging a Small Task Corpus In Section 4.3, we propose a simple method for leveraging the potentially small corpus of a target task to build good in-domain representations. Among the several variants that we have tested, the best-performing one consists in using a general-domain ELMo model which we further pre-train on the task corpus. We then also train static representations with fastText on this task corpus and combine both representations. While a simple concatenation of both embeddings works well, we show, in the specific context of using ELMo, that the static vectors can be injected directly in the internal linear combination of the model to reach results that are on par with medical variants of ELMo.

General WordPieces in a Specialized Domain We construct an in-domain WordPiece vocabulary and show that general-domain WordPieces lead to a seemingly less meaningful tokenization when used on texts from a specialized domain. While it is not obvious that this would necessarily have an impact on the downstream performance, it is still interesting to be aware of such biases when re-training a general BERT model on a specialized corpus.

Re-training vs. Training from Scratch We train parallel versions of BERT: one using a general WordPiece vocabulary, which we train on a general corpus before re-training on a corpus from the specialized domain; and another using a specialized vocabulary, which we train directly on specialized corpora. We show that while the medical model is initially superior to the general one, both models ultimately perform at the same level, despite a slight edge for the fully specialized version. As a result, and given the cost of training models from scratch, we suggest building specialized models by re-training from a general version.

CharacterBERT We propose a new variant of BERT which drops the WordPiece system altogether and uses a CharacterCNN borrowed from ELMo instead. We show that this variant improves over versions of BERT that are trained in similar conditions and demonstrate that it can be more robust to misspellings while at the same time being word-level and open-vocabulary.

Encoding Knowledge Bases into Dense Vectors As a preliminary step before attempting to enhance word representations using external knowledge, we have extracted graphs of concepts from the UMLS meta-thesaurus and subsequently trained knowledge representations using node2vec for two medical vocabularies: MeSH and SNOMED CT.

Combining Word and Knowledge Representations We demonstrate that a simple concatenation can lead to improvements when used to combine existing word representations, either static or contextual, with node2vec knowledge embeddings. Moreover, we extend this framework to the internal representations of transformer-based models and propose Knowledge Injection Modules (KIM), which effectively incorporates external knowledge within the hidden states of BERT-like models. Nevertheless, we argue that selecting the appropriate information to inject is key and that it should remain the primary focus when attempting to specialize existing representations with external knowledge.

Code and Pre-trained Models We compile all the code we have produced throughout this work and share it along with our pre-trained models to benefit the

NLP community and assist future research on the topic of specializing word embeddings using in-domain corpora and knowledge bases[1].

7.3 Future Directions

We review a number of ideas related to the topics we have covered so far and that may be worth exploring in future work:

Corpus Size vs. Similarity Given the seemingly diminishing returns of increasing corpus size while keeping domain similarity constant, it is important to be able to say what constitutes an optimal corpus size for a given task. However, the underlying dynamics are probably more complex than it seems and may involve other aspects such as vocabulary coverage and co-occurrence frequencies. In fact, given that most embedding methods rely more or less directly on co-occurrences to learn word representations, an interesting future direction would be, given a large in-domain corpus, to programmatically sample sub-corpora while controlling both the vocabulary coverage and co-occurrence frequencies to rigorously analyse the impact of these parameters. In this context, vocabulary coverage would indirectly represent domain similarity while co-occurrence frequencies would indirectly represent corpus size.

Leveraging a Small Task Corpus We have seen how off-the-shelf ELMo representations can be used along with a small task corpus to produce good in-domain embeddings. Future directions may attempt to apply the same idea to more recent transformer-based models. One approach is to simply use concatenation (as we have done for ELMo). More elaborated solutions may attempt to adapt modern meta-embedding approaches, which would expect two sets of static vectors, to the case of transformer-based models on one hand and static vectors on the other. On an unrelated note, other directions could investigate how data selection can be applied to generally available corpora (e.g. Wikipedia) to augment the target task corpus. Interestingly, this idea could be used in conjunction with the meta-embedding approach to produce potentially greater improvements.

General WordPieces in a Specialized Domain Given the ultimately satisfactory results of medical versions of BERT that are constructed via re-training from a general model, it is clear that despite the seemingly worse tokenization of the general WordPiece vocabulary when used on specialized texts, the final model is able to nonetheless construct relevant internal representations. This may mean that the first layers of the models are partially

[1] https://github.com/helboukkouri/phd-code

used for contextualizing seemingly meaningless WordPiece representations into more meaningful vectors. An interesting research topic would be investigating whether this contextualization happens then understanding its mechanics. Once understood this may shed some light on ways to construct good character-level transformer-based models which would only use single character (or byte) WordPieces.

Re-training vs. Training from Scratch While our experiments compare two standard approaches (i.e. re-training vs. training from scratch), they fail to highlight the impact of using a general vocabulary in a specialized domain. One supplementary experiment which would show this effect more clearly would be training a model from scratch on a specialized corpus but using a general vocabulary. Comparing this configuration to the fully specialized version would allow to have a definitive answer on whether this affects downstream performance or not.

CharacterBERT Given that the BERT model that was trained parallel to CharacterBERT performs below original BERT, we have reasons to think that our versions of CharacterBERT can be further optimized. Moreover, there are many BERT-like models that can be converted into character-based versions using the same principle (i.e. dropping the WordPieces and using a CharacterCNN instead). Finally, since CharacterBERT does not rely on a WordPiece vocabulary, an interesting direction would be to train a multi-lingual version of the model as well as to investigate its zero-shot capabilities in a cross-lingual context.

Encoding Knowledge Bases into Dense Vectors We have used node2vec in an effort to simplify our overall knowledge injection method. Future research directions may focus on using actual knowledge base embedding techniques such as TransE (and also more recent alternatives), that leverage knowledge triples to produce entity representations while considering all available kinds of relations. Another orthogonal direction would be to leverage the knowledge triples directly, however that would imply adopting a different knowledge injection approach. Nevertheless, we can imagine training word representations in a multi-task fashion, using both the original pre-training objective and an additional retrofitting-style constraint.

Combining Word and Knowledge Representations Concatenation is a strong baseline for meta-embeddings, however, there are many more alternatives which are likely to produce better results. Future directions may investigate other meta-embedding methods such as the previously mentioned Word Prisms which could leverage knowledge embedding methods that produce

relation-specific entity representations. Another less obvious direction, which may be tedious but is nonetheless likely to benefit our method, is better concept normalization (i.e. entity linking). In fact, we recall that we used a naive baseline for normalization which consists in projecting the UMLS terms onto our corpora. As a result, there have been multiple instances of out-of-concept tokens that were wrongly linked to concepts and vice versa. Another related direction is to investigate the impact of casing on this normalization as, for instance, the medical domain has multiple acronyms that may be confused with common word when converted to lowercase (e.g. "AS" for "Aortic Stenosis").

Appendix A

Neural Networks and Deep Learning

In this appendix, we provide a bit of technical background for understanding the main neural architectures that are commonly used in Natural Language Processing in general, as well as in the context of this thesis in particular. Namely, in Section A.1, we go over some basic components and concepts that occur when dealing with such neural models. These are, for instance, the usual activation functions, neural layers (e.g. convolutional, recurrent layers, etc) and regularization components. Then, in Section **??**, we delve more specifically into the architecture of two models that play an important role in the work presented in this manuscript, namely: ELMo (Peters et al., 2018a) and BERT (**?**).

A.1 Fundamental Concepts

Neural networks have varied greatly in both form and implementation since the original instances developed during the 1950's. In fact, the first widely recognized type of neural networks was the *perceptron* (a.k.a. *artificial neuron*), a linear model with parameters $\boldsymbol{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$ that could produce a binary signal according to its input features $\boldsymbol{x} \in \mathbb{R}^d$:

$$\textsc{Perceptron}(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \boldsymbol{w} \cdot \boldsymbol{x} + b > 0, \\ 0 & \text{otherwise} \end{cases} \tag{A.1}$$

However, this simple prototype lacked expressive power and was unable to model basic phenomena such as the XOR function (Minsky and Papert, 1988).

In modern Deep Learning, the neural architectures that are commonly used are usually far more complex than a single neuron and can be utilized to solve a wide range of problems, often with state-of-the-art performance. To achieve such power, the basic idea behind these architectures is to compose multiple *layers* that successively transform the input information into a final set of features that can be

mapped to a desired output. The field of Machine Learning that is interested in coming up with such nested neural network architectures is called Deep Learning (Schmidhuber, 2015; Salakhutdinov, 2014).

Most often than not, neural networks consist of multiple layers that are *stacked* on top of each other. There is a rich variety of layers in the Deep Learning literature which we do not attempt to cover here. However, we go over some fundamental components which usually appear in the context of NLP and specifically in the work presented in this manuscript.

A.1.1 Activation Functions

The power of neural networks stems in part from their ability to model complex non-linear functions. In order to achieve this, these models rely on so-called *activation functions* that introduce non-linearities in the otherwise linear system.

Sigmoid Although there is a larger family of sigmoid functions, the term "sigmoid" usually refers in Deep Learning to the standard logistic function:

$$\sigma(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{1+e^x} \tag{A.2}$$

$$\frac{\mathrm{d}}{\mathrm{d}x}\sigma(x) = \frac{e^x}{(1+e^x)^2} = \sigma(x)\big(1-\sigma(x)\big) \tag{A.3}$$

This is a strictly increasing function that is used, in practice, to bound a layer's output within the range $(0, 1)$ while introducing some level of non-linearity.

Tanh The hyperbolic tangent function can be defined in different ways. One common way is using the hyperbolic sine and cosine functions, which are respectively defined as the odd and even parts of the exponential function:[1]

$$\tanh x = \frac{\sinh x}{\cosh x} = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{A.4}$$

$$\frac{\mathrm{d}}{\mathrm{d}x}\tanh(x) = \frac{\cosh^2(x) - \sinh^2(x)}{\cosh^2(x)} = 1 - \tanh^2(x) \tag{A.5}$$

However, a more practical way to look at it is as a sigmoid analogue that allows for negative values. In fact, we have the following relation:

$$\tanh(x) = \sigma(2x) - \frac{e^{-2x}}{1+e^{-2x}} = 2\sigma(2x) - 1 \in (-1, 1) \tag{A.6}$$

[1] You can read more about odd-even decomposition in this Wikipedia article.

ReLU One common problem with the previous activation functions is the so-called "vanishing gradient" issue (Hochreiter et al., 2001). This problem occurs during back-propagation when applying the chain-rule throughout the network and multiplying multiple small valued gradients[2], effectively killing the back-propagation signal and preventing the network from training properly. An alternative to these "saturating non-linearities" that is both mathematically and biologically motivated is the ReLU function (Hahnloser et al., 2000; Hahnloser and Seung, 2000):

$$\text{ReLU}(x) = x^+ = \max(0, x) \tag{A.7}$$

ReLU functions are more computationally efficient and have been shown to improve neural network training (Glorot et al., 2011; Maas et al., 2013).

GELU This is a smooth variant of ReLU that often performs better in practice by basically weighting input values according to how large they are to other inputs (Hendrycks and Gimpel, 2016). In the context of GELU, input values are assumed to be distributed according to a standard normal distribution[3] which leads to:

$$\text{GELU}(x) = x \cdot \mathbb{P}_{\mathcal{N}(0,1)}(X < x) = x \cdot \Phi(x) \approx x \cdot \sigma(1.702x) \tag{A.8}$$

where Φ is the cumulative function of the normal distribution $\mathcal{N}(0, 1)$.

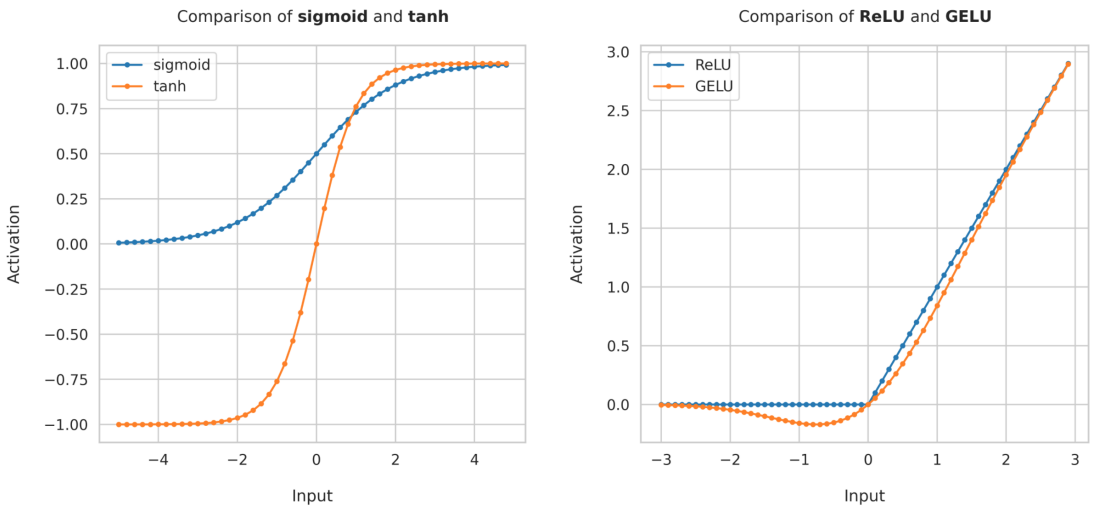

Figure A.1: Comparison of various activation functions.

[2]The absolute values of the tanh and sigmoid derivatives always fall in the open interval $(0, 1)$.
[3]This is a reasonable assumption in many cases, especially when using normalization layers.

A.1.2 Regularization & Normalization

Other important aspects of neural network architectures are regularization and normalization. In fact, as large neural layers are stacked to achieve greater expressive power, neural networks become both more prone to overfitting (i.e. over adapt to the training data and do not generalizing well to unseen data) and more unstable (i.e. do not converge properly to an "optimal" model).

Dropout To mitigate overfitting in large neural networks, Dropout (Srivastava et al., 2014) randomly turns off a subset of neurons (i.e. multiplies the outputs by zero) at each training iteration (see Figure A.2). Conceptually, this is similar to training by alternating between multiple smaller models that are less likely to overfit instead of a single larger one. At test time, however, all neurons are activated to allow the model to use all of its activations.

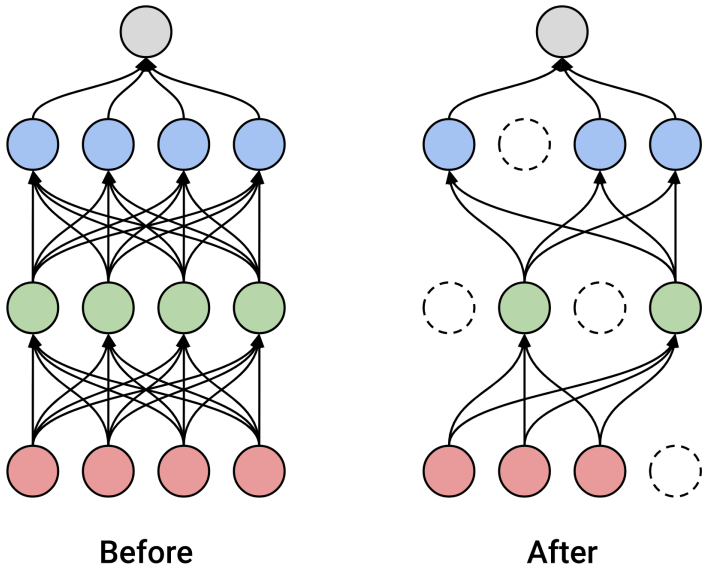

Figure A.2: Comparison of a neural network with two hidden layers before and after applying Dropout (Srivastava et al., 2014).

Let $\boldsymbol{y} \in \mathbb{R}^n$ be the output of some given neural network layer and $p \in (0, 1)$ the chosen dropout probability, then:

$$\text{Dropout}(\boldsymbol{y}) = \begin{cases} \boldsymbol{y} \cdot \boldsymbol{z} & \text{at training time,} \\ \boldsymbol{y} & \text{at inference time} \end{cases} \tag{A.9}$$

where $\boldsymbol{z} = [z_1, z_2, \ldots, z_n] \in \{0, 1\}^n$ is re-sampled from a Bernouilli $\mathcal{B}(p)$ distribution for each mini-batch of training data.

BatchNorm As the parameters of a neural network's layers are updated during back-propagation, their output distributions change as well. This makes the end-to-end training of deep neural networks unstable as subsequent layers may not expect such changes in input distributions. To address this so-called "internal covariate shift",[4] Batch Normalization was proposed to shift and scale hidden activations to zero mean and unit variance (Ioffe and Szegedy, 2015):

$$\textsc{BatchNorm}(\boldsymbol{x}_i) = \gamma \cdot \frac{\boldsymbol{x}_i - \mathbb{E}_{\text{minibatch}}(\boldsymbol{x}_i)}{\sqrt{\mathbb{V}_{\text{minibatch}}(\boldsymbol{x}_i) + \epsilon}} + \beta \tag{A.10}$$

where $\boldsymbol{x}_i$ are the i-th activations produced by a given layer over the mini-batch, $\mathbb{E}_{\text{minibatch}}$ and $\mathbb{V}_{\text{minibatch}}$ are the estimated mean and variance over the mini-batch, ϵ is a small positive value for numerical stability and γ, β are respectively scaling and bias parameters to recover the model's expressive power.

LayerNorm The main drawback of BatchNorm is the fact that it requires larger batch sizes to achieve a good estimation of the sample mean and variance. This makes it less suited for situations with small batch sizes like Online Learning (i.e. having a single example per batch) or training with gradient accumulation. Instead of normalizing each activation using the estimated mean and variance over all examples in a mini-batch, Layer Normalization (Ba et al., 2016) normalizes each activation using parameters estimated over all other activations at the example-level, thus removing the dependency to the batch size:

$$\textsc{LayerNorm}(\boldsymbol{x}) = \gamma \cdot \frac{\boldsymbol{x} - \mathbb{E}_{\text{example}}(\boldsymbol{x})}{\sqrt{\mathbb{V}_{\text{example}}(\boldsymbol{x}) + \epsilon}} + \beta \tag{A.11}$$

where $\boldsymbol{x}$ are the activations produced by a given layer for a specific example, $\mathbb{E}_{\text{example}}$ and $\mathbb{V}_{\text{example}}$ are the estimated mean and variance over $\boldsymbol{x}$ and ϵ, γ, β play the same role as in BatchNorm.

A.1.3 Common Neural Network Layers

In this section we introduce different types of neural network layers that are used in architectures relevant to the topics of this thesis.

[4] BatchNorm was first thought to improve neural network training by reducing the Internal Covariate Shift, however, this seems not to be the case. Instead, batch normalization was shown to smooth the optimization landscape which is now thought to be the main reason for its performance (Santurkar et al., 2018).

DENSE Layers These are arguably the fundamental building blocks of all neural networks. DENSE layers (a.k.a *linear* or *fully connected* layers) apply a simple affine transformation between spaces of usually different dimensions, followed by an optional non-linear activation function. For example, given an input size n and an output size m, a DENSE layer with sigmoid activation applies the following transformation:

$$\text{DENSE}_\sigma(\boldsymbol{x}) = \sigma(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}) \tag{A.12}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ are the input features and $\boldsymbol{A} \in \mathbb{R}^{m \times n}$, $\boldsymbol{b} \in \mathbb{R}^m$ are respectively the weight and bias parameters of the dense layer.

HIGHWAY Layers Neural network layers are usually stacked into deep architectures. In these situations, training can be difficult as the backpropagation signal may have a hard time traveling through the entire depth of the model. For this reason, HIGHWAY layers (Srivastava et al., 2015) introduce a "gating" mechanism that allows the model to smoothly vary a layer's behaviour between applying the intended transformation and simply passing through the input features to the next layer. Given an original layer with transformation f, applying a HIGHWAY transformation consists in:

$$\begin{cases} \text{HIGHWAY}(\boldsymbol{x}, f) = g(\boldsymbol{x}) \cdot f(\boldsymbol{x}) + \big(1 - g(\boldsymbol{x})\big) \cdot \boldsymbol{x} \\ g(\boldsymbol{x}) = \sigma(\boldsymbol{A}_g \boldsymbol{x} + \boldsymbol{b}_g) \in (0, 1) \end{cases} \tag{A.13}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ are the input features and g is the gating mechanism, which is itself a DENSE layer with sigmoid activation and parameters $\boldsymbol{A}_g \in \mathbb{R}^{n \times n}$, $\boldsymbol{b} \in \mathbb{R}^n$.

Convolutional Layers These are the main building blocks of Convolutional Neural Networks (CNN) (LeCun et al., 1989) that were originally developed in the context of computer vision (Fukushima and Miyake, 1982) before being applied to other types of data like texts as well (Kim, 2014b). Generally speaking, Convolutional layers rely on a set of (usually) small 2D *filters* (a.k.a *kernels*) to transform in input image into multiple feature maps (one for each filter). These feature maps are obtained by scanning the input image and computing dot products between local areas of the image and each filter through a process called convolution,[5] usually followed by a RELU activation function. In the context of NLP, if each word in a sentence of size n is represented as a vector of size m, then the input can be seen as an $n \times m$ matrix and the same process[6] can be applied.

[5] Follow this link for a visual representation of the convolution operation.

[6] The only difference is that instead of scanning the input along both image dimensions, all local regions of the text input span over the entire embedding dimension and the matrix is

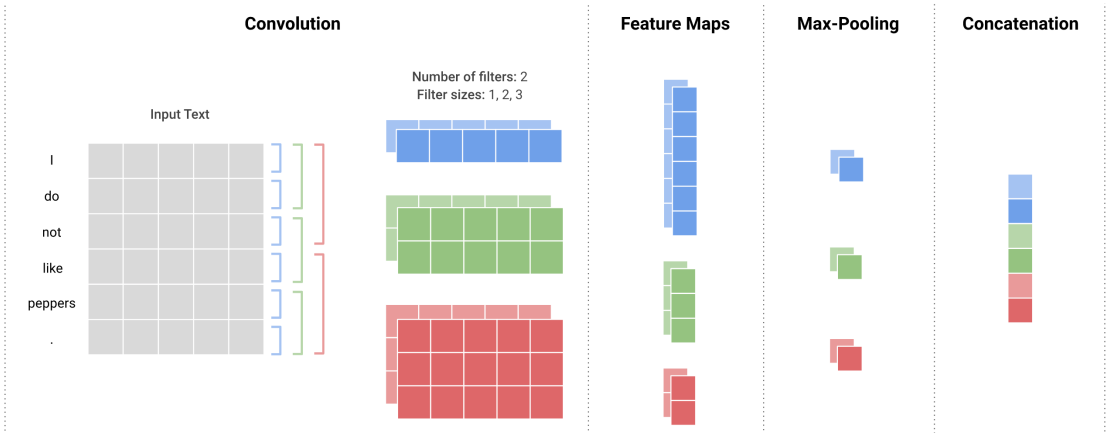

Figure A.3: Illustration of a convolutional layer converting text into features.

Convolutional layers are useful to detect local patterns—as well as more global and abstract patterns by stacking multiple such layers —in large structures such as images (by looking at neighbouring pixels) or sentences (by looking at neighbouring words or characters). In practice, we often extract feature maps using different convolutional layers before applying some type of pooling followed by concatenation to recover a final feature vector (see Figure A.3).

Recurrent Layers These are the go-to type of layers when dealing with sequential data where the information at a specific time-step benefits from the history of all previous time-steps. In order to condition each element of the input (e.g. a sequence of words) on the value or state of previous elements, Recurrent Neural Networks (RNNs) feed back the activations of each element as additional inputs when computing the output for the next element (see Figure A.4). Such RNNs are supposed to be able to keep a memory of previous inputs however, in practice, they seem to only be able to keep a short-term memory. To address this issue, Long-Short Term Memory networks (or LSTMs) (Hochreiter and Schmidhuber, 1997) introduce a type of explicit memory that can be accessed and altered using a number of different gates. Let $\{\boldsymbol{x}_t\}_{0 \leq t \leq n}$ a sequence of input vectors (e.g. word embeddings), then the LSTM memory (a.k.a *cell state*) is:

$$\boldsymbol{c}_t = \boldsymbol{f}_t \cdot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \cdot \tilde{\boldsymbol{c}}_t \tag{A.14}$$

where $\boldsymbol{c}_{t-1}$ is the memory at the previous time-step, $\tilde{\boldsymbol{c}}_t$ is a candidate update to the memory and $\boldsymbol{f}_t$, $\boldsymbol{i}_t$ are respectively the forget and input gates which, together, determine how the cell state is altered. Given the current memory $\boldsymbol{c}_t$, the output

scanned along the sentence dimension only. This is often referred to as a 1d convolution.

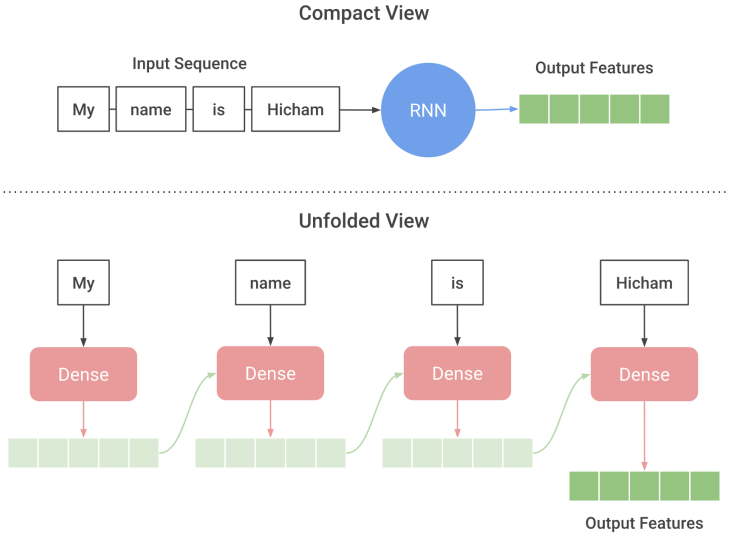

Figure A.4: A basic RNN unfolding as a sequence of dense layers.

of the RNN at time-step t is:

$$\boldsymbol{h}_t = \boldsymbol{o}_t \cdot \tanh(\boldsymbol{c}_t) \tag{A.15}$$

where $\boldsymbol{o}_t$ is the output gate. All three gates $\boldsymbol{i}_t$, $\boldsymbol{f}_t$ $\boldsymbol{o}_t$ as well as the candidate update $\tilde{\boldsymbol{c}}_t$ are computed using the current *input* $\boldsymbol{x}_t$ as well as the previous *output* $\boldsymbol{h}_{t-1}$ using the following formulas:

$$\begin{aligned}
\boldsymbol{i}_t &= \sigma(\boldsymbol{W}_i x_t + \boldsymbol{U}_i \boldsymbol{h}_{t-1} + \boldsymbol{b}_i) \\
\boldsymbol{f}_t &= \sigma(\boldsymbol{W}_f \boldsymbol{x}_t + \boldsymbol{U}_f \boldsymbol{h}_{t-1} + \boldsymbol{b}_f) \\
\boldsymbol{o}_t &= \sigma(\boldsymbol{W}_o \boldsymbol{x}_t + \boldsymbol{U}_o \boldsymbol{h}_{t-1} + \boldsymbol{b}_o) \\
\tilde{\boldsymbol{c}}_t &= \tanh(\boldsymbol{W}_{\tilde{c}} \boldsymbol{x}_t + \boldsymbol{U}_{\tilde{c}} \boldsymbol{h}_{t-1} + \boldsymbol{b}_{\tilde{c}})
\end{aligned} \tag{A.16}$$

where $\boldsymbol{W}_{(\cdot)}$, $\boldsymbol{U}_{(\cdot)}$ are respectively weight parameters for the current input and previous output, and $\boldsymbol{b}_{(\cdot)}$ are the bias parameters, all for the different gates: i (input), o (output), f (forget) and $\tilde{c}$ (cell state update). Note that all gates produce non-negative values with the exception of the cell state update.

Transformer Layers These are the main components of recent state-of-the-art models like BERT. Transformer layers (Vaswani et al., 2017) consist in a relatively complex architecture that has been developed in the context of sequence-to-sequence tasks (e.g. for Machine Translation), and that originally involved both an encoder and a decoder module (see Figure A.5). However, due

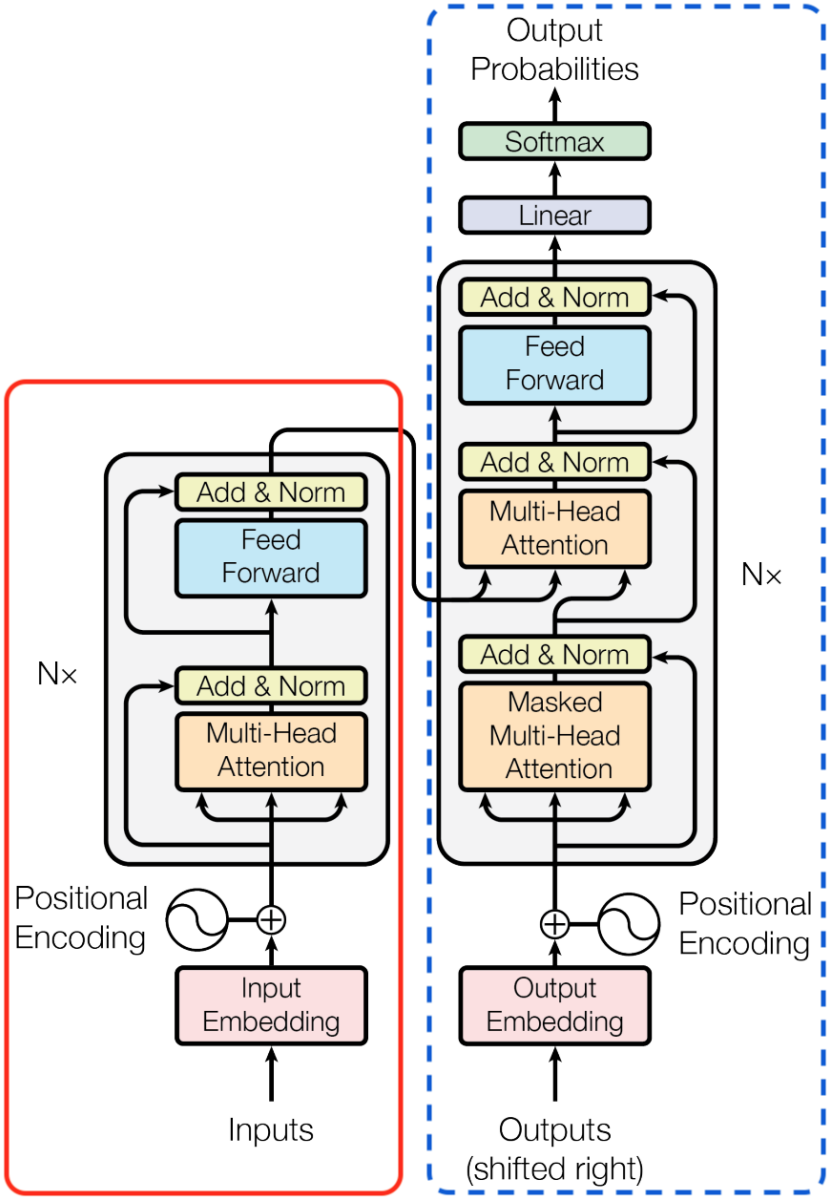

Figure A.5: The TRANSFORMER architecture. Original figure from Vaswani et al. (2017). The encoder module is surrounded by a red line and the decoder module is highlighted with a blue dash-line.

to BERT and other subsequent transformer-based models relying exclusively on the encoder part, the term "TRANSFORMER" has become somewhat synonymous with "TRANSFORMER encoder". Loosely speaking, TRANSFORMER layers can be seen as an alternative to RNNs that is able to condition each element from the input sequence in a truly bi-directional fashion—as opposed to concatenating two uni-directional representations (e.g. Bi-LSTM)—and in a way that lends itself more easily to parallel computing. Moreover, this architecture allows every token to directly attend to all other tokens of the input sequence, which can be contrasted with keeping a "memory" that indirectly gives information about the input's content as is usual with recurrent layers.

At the heart of every TRANSFORMER lies a mechanism called MULTI-HEAD ATTENTION that is responsible for "contextualizing" each element of the input sequence. This mechanism relies on three sets of vectors called queries Q, keys K and values V, which, put simply, are used to compute for each input element a weighted sum of the values $\{v_t\}_{1\leq t\leq n}$ using weights that reflect how similar the current element's query q is to all the other elements' keys $\{k_t\}_{1\leq t\leq n}$. In practice, the query, key and value vectors are learned as projections of the input representations, and the overall attention computation is repeated for multiple "heads", each consisting in a different projection of the input vectors, before all heads' outputs are concatenated and projected down to produce the final sequence of contextualized representations. If we stack all d-dimensional input vectors into a matrix $\boldsymbol{X}$, then the MULTI-HEAD ATTENTION output can be expressed as:

$$\begin{cases} \text{MULTI-HEAD ATTENTION}(\boldsymbol{X}) = [\text{HEAD}_1, \text{HEAD}_2, \ldots, \text{HEAD}_h] \cdot \boldsymbol{W}^o \\ \text{HEAD}_i = \text{SOFTMAX}\left(\dfrac{\boldsymbol{Q}_i \cdot \boldsymbol{K}_i^T}{\sqrt{d_k}}\right) \cdot \boldsymbol{V}_i, \quad i \in \{1, \ldots, n\} \\ Q_i = \boldsymbol{X}\boldsymbol{W}_i^Q, \quad K_i = \boldsymbol{X}\boldsymbol{W}_i^K, \quad V_i = \boldsymbol{X}\boldsymbol{W}_i^V \end{cases} \tag{A.17}$$

where $\boldsymbol{W}_i^Q,\ \boldsymbol{W}_i^K,\ \boldsymbol{W}_i^V\ \in \mathbb{R}^{d\times d_k}$ are respectively the projection matrices for the queries, keys, values of the i-th attention head, and W^o is the final projection matrix of the attention mechanism.

While the MULTI-HEAD ATTENTION mechanism is perhaps the most important aspect of TRANSFORMER layers, there are various other components that contribute to the architecture's efficacy such as residual connections, feed-forward layers, normalization layers, etc (see Figure A.5). Another such component that is crucial for the proper functioning of these layers is the use of *positional encodings*. In fact, as the position information does not play a role in the attention computation process (i.e. the same vector in different positions would produce the same output), transformer-based architectures need to bias their input representations

using vectors that hint to each token's position. In the original TRANSFORMER by Vaswani et al. (2017), the authors proposed to add to each input token's vector a d-dimensional sinusoid-based representation that would depend the token's position p:

$$\text{POSENCODING}(p) = \left[\ldots, \sin\left(\frac{p}{10000^{2i/d}}\right), \cos\left(\frac{p}{10000^{(2i+1)/d}}\right), \ldots \right] \tag{A.18}$$

The authors also experimented with learned positional encodings and showed that both versions achieve similar results. In subsequent models such as BERT, the sinusoid-based encodings were dropped in favour of learned positional embeddings.

A.2 Contextual Language Models

In recent years, there have been an important shift in the way textual information is represented in Natural Language Processing tasks, specifically in word embedding models which have transitioned from a static paradigm, where each word is assigned a single vector; to a contextual paradigm, where the same word may have different representations according to its context. In this section we provide an overview of two main architectures responsible for producing such context-dependent word representations, namely: ELMO Peters et al. (2018a) and BERT (**?**).

A.2.1 ELMo

Embeddings from Language Models, or ELMO for short, is a neural architecture that was conceived in an attempt to produce context-dependent word representations. Specifically, the model uses an architecture based on bi-directional recurrent layers that is first trained on a language modeling task (i.e. predicting the next word in a sentence). Then, the language modeling head is discarded and the models internal representations are combined to form contextualized word representations.

ELMO's exact architecture comprises three modules: a CHARACTERCNN then two successive Bi-LSTMs (see Figure A.6). The CHARACTERCNN is responsible of converting input words into initial context-independent representations before they can be contextualized using the downstream recurrent layers. The exact process involves converting each word into a sequence of characters, embedding each character, using a set of CNN layers to compute convolutions over the sequence then combining the CNN outputs into a final vector that can be projected down to a desired size.[7]. The two following Bi-LSTM layers then rely on this initial

[7]To avoid redundancy, we do not detail the inner-workings of the CHARACTERCNN which are further discussed in Section 5.3.1.

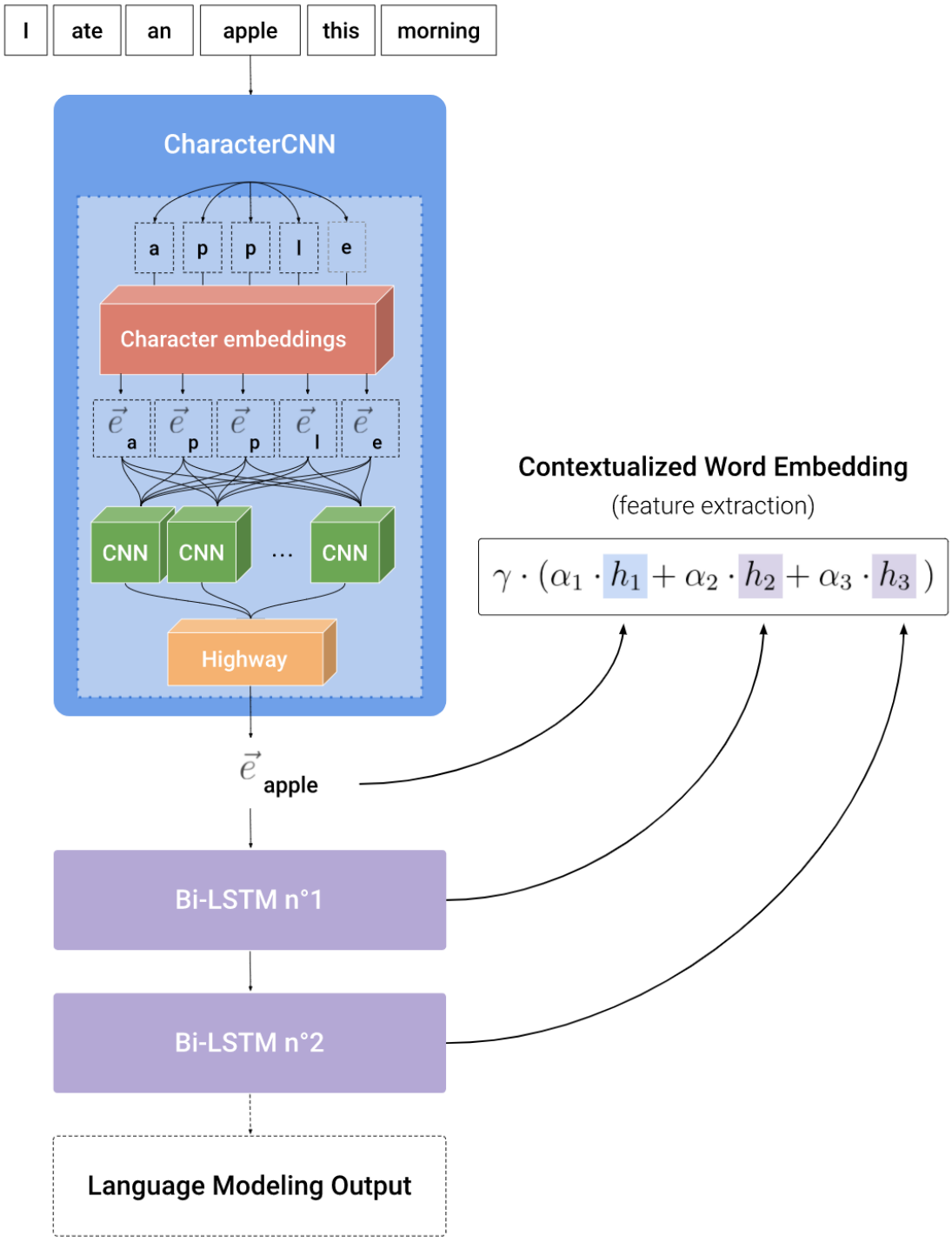

Figure A.6: Overview of ELMo's architecture. The model is trained on language modeling then used as a feature extractor on various downstream tasks.

embedding to produce a contextualized representation which can serve as a basis for the language modeling pre-training task. Ultimately, after the pre-training is complete, ELMo can be used as a contextual embedding generator (i.e. feature extraction), producing for any word or token w, features in the form of:

$$\text{ELMo}(w) = \gamma \cdot (\alpha_1 \cdot h_1 + \alpha_2 \cdot h_2 + \alpha_3 \cdot h_3) \tag{A.19}$$

where $\gamma, \alpha_1, \alpha_2$ and α_3 are tunable task-specific coefficients and h_1, h_2 and h_2 respectively the CharacterCNN, 1^{st} and 2^{nd} Bi-LSTM representations. It is important to note however that while the linear combination weights α_1, α_2 and α_3 can take any real value, they actually undergo a softmax transformation that converts them in to positive values that sum to 1.

A.2.2 BERT

Bidirectional Encoder Representations from Transformers , or BERT for short, is yet another neural language model that is able to produce context-dependent word representations. However, contrary to ELMo, which relies on a CharacterCNN and two recurrent layers to generate such representations, BERT adopts a different approach that consists of a WordPiece tokenization system (Wu et al., 2016) coupled with Transformer layers.

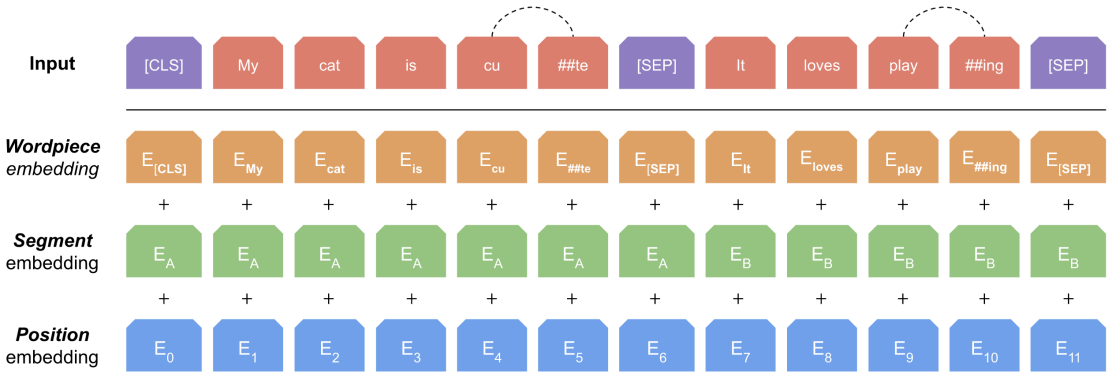

Figure A.7: Overview of BERT's input. Both input sequences are joined using special tokens [CLS] and [SEP], then the WordPiece embedding of each token looked up then biased using segment and position embeddings.

The WordPiece tokenization system relies on a predefined set of subwords (called WordPieces) which are learned on the same corpus that is used for pre-training. Given this subword vocabulary, any incoming token that is out-of-vocabulary is decomposed into the fewest number of subwords with a possible

fallback at the level of the characters. As a result, instead of using a CharacterCNN module like ELMo, BERT can simply train a WordPiece embedding matrix, split potential OOVs into subwords and embed each unit accordingly. BERT being a model that relies purely on attention-based layers, it is by default not able to encode information about the position of a token in a sentence as would otherwise a recurrent layer (i.e. for a Transformer layer, a sequence and the same sequence but shuffled produce the same representations). To deal with this issue, the initial static WordPiece representations are enriched with position and segment embeddings (see Figure A.7).

After the initial embedding layer, the static WordPiece representations go through a series of Transformer layers (see definition in previous section) eventually resulting in contextual subword representations that can be used to encode texts of interest. However, contrary to ELMo which is used in a feature extraction fashion (i.e. generating representations that serve as "fixed"[8] features for downstream layers), BERT is used as an encoder and is traditionally fine-tuned in an end-to-end fashion with the appropriate task-specific heads.

One last characteristic of BERT is its pre-training procedure. In fact, while more traditional language models would be trained on a task consisting in predicting the next words in a sentence, BERT is trained on Masked Language Modeling which takes advantage of the bidirectional nature of the model and aims to predict masked words in a sentence using both the left and right contexts. Moreover, this model is also trained on a second task called Next Sentence Prediction where pairs of sentences have to be classified as either successive or random. This task however is widely thought to be detrimental to the overall quality of the model and is ignored in subsequent variants of BERT such as RoBERTa (Liu et al., 2019).

[8]While the ELMo architecture is fixed, the linear combination weights remain trainable.

Appendix B

Knowledge Injection Architectures

Since literal model descriptions can be ambiguous, we provide complementary illustrations for each of the model architectures from Section 6.3.2.

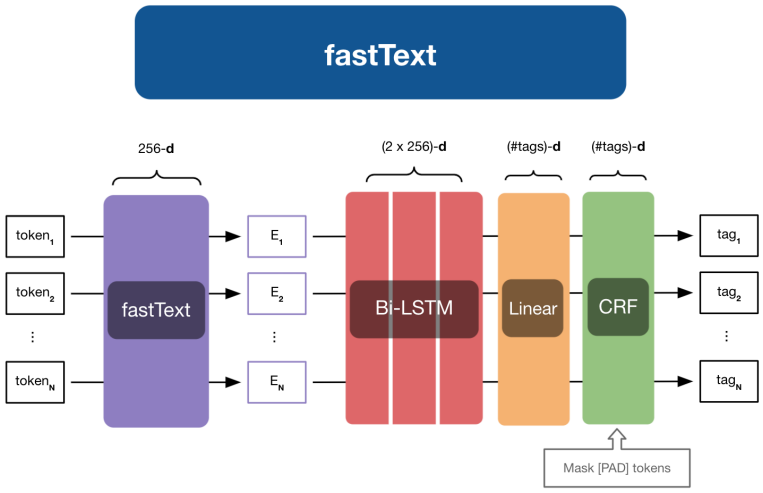

Figure B.1: Knowledge Injection for NER: fastText.

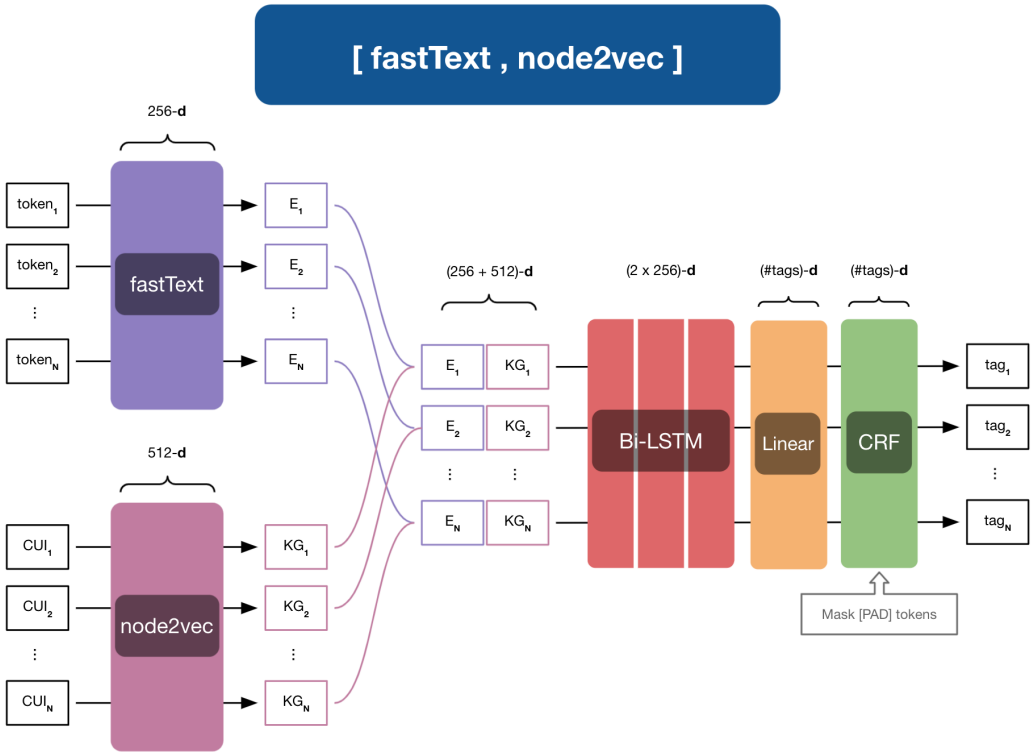

Figure B.2: Knowledge Injection for NER: [fastText, node2vec].

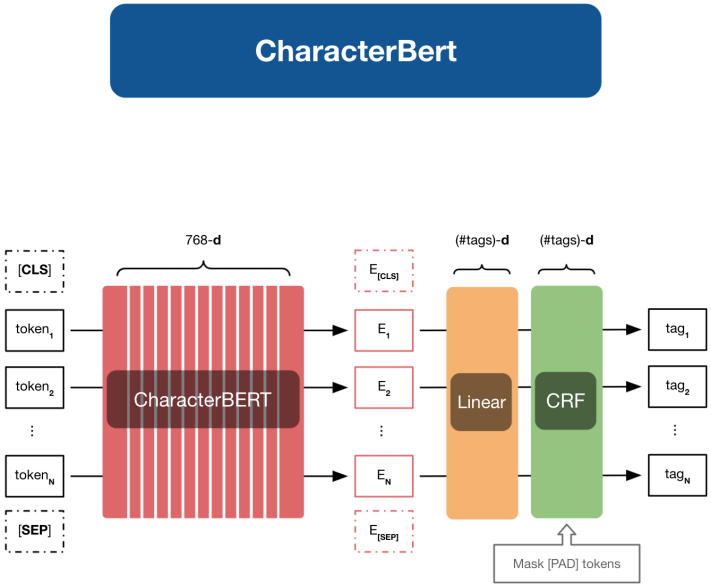

Figure B.3: Knowledge Injection for NER: CharacterBERT.

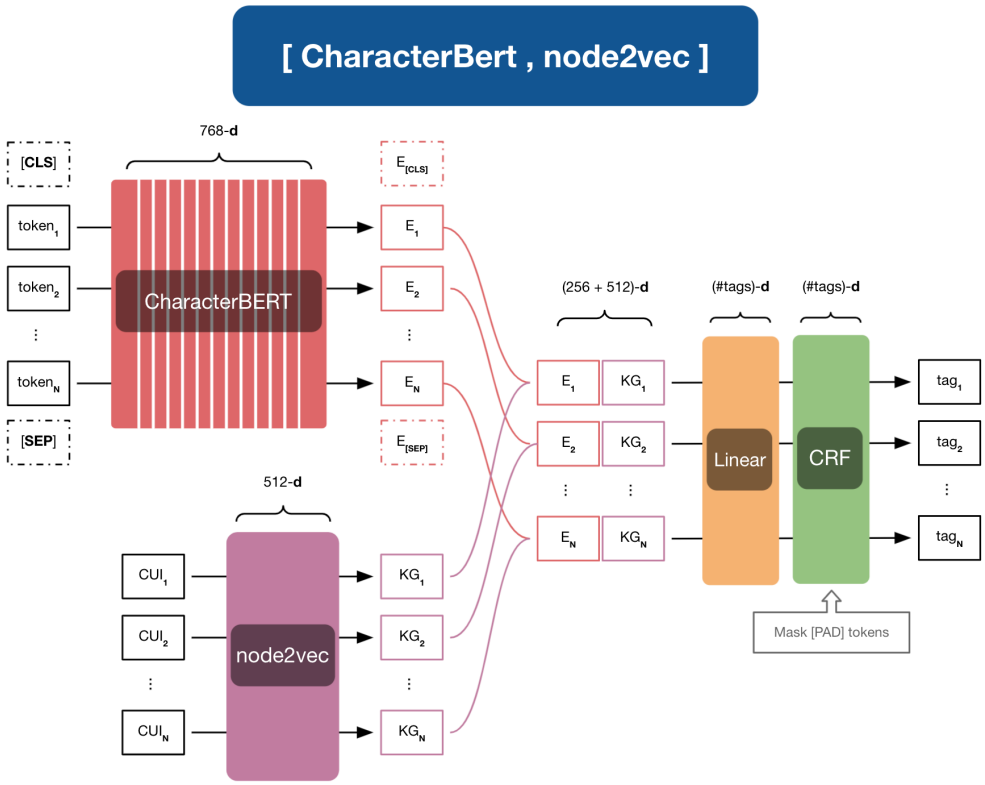

Figure B.4: Knowledge Injection for NER: [CharacterBERT, node2vec].

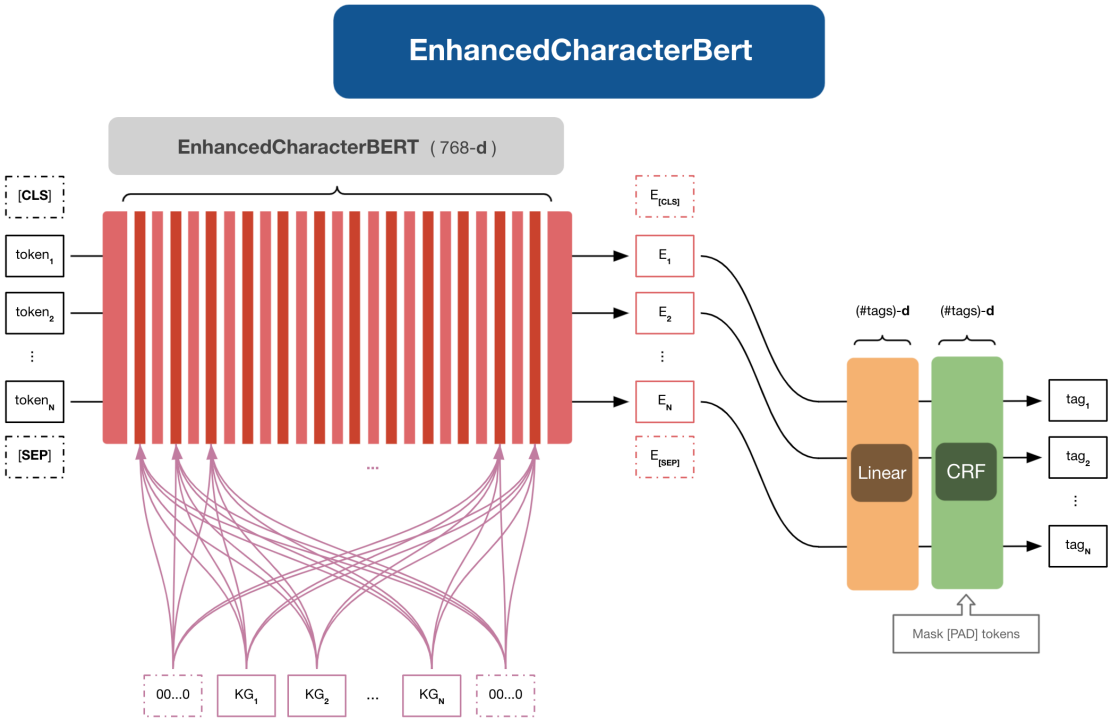

Figure B.5: Knowledge Injection for NER: EnhancedCharacterBERT.

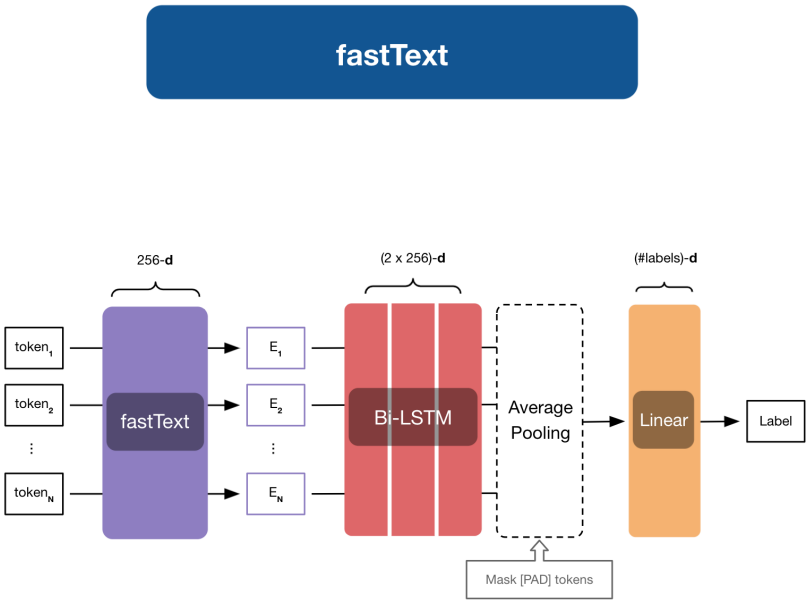

Figure B.6: Knowledge Injection for Classification: FASTTEXT.

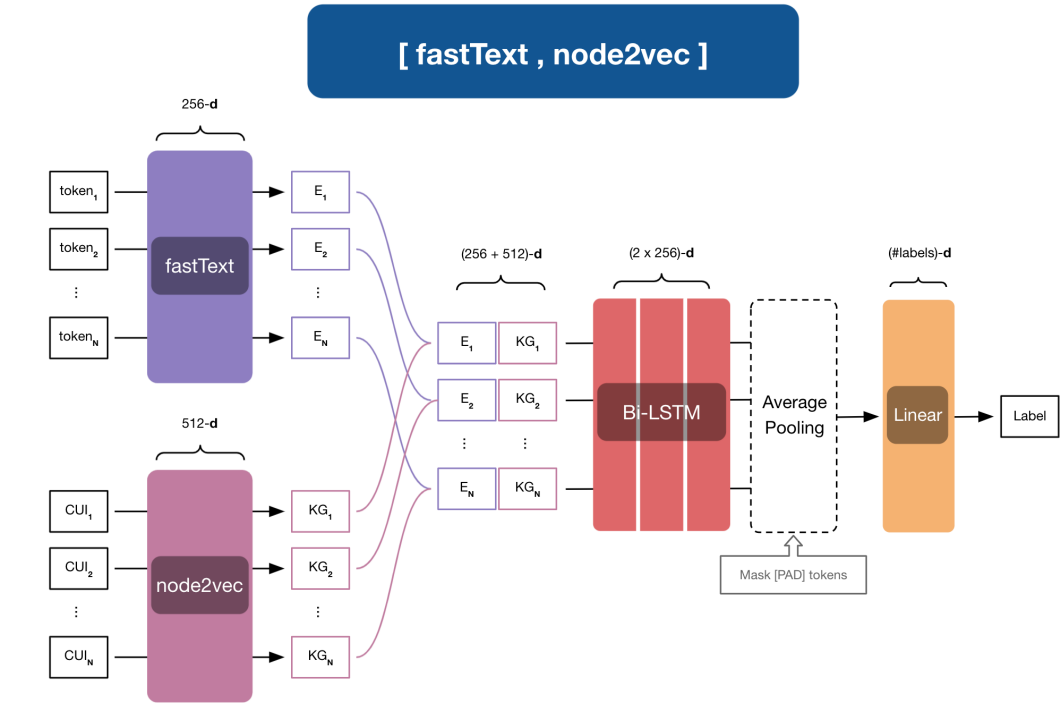

Figure B.7: Knowledge Injection for Classification: [fastText, node2vec].

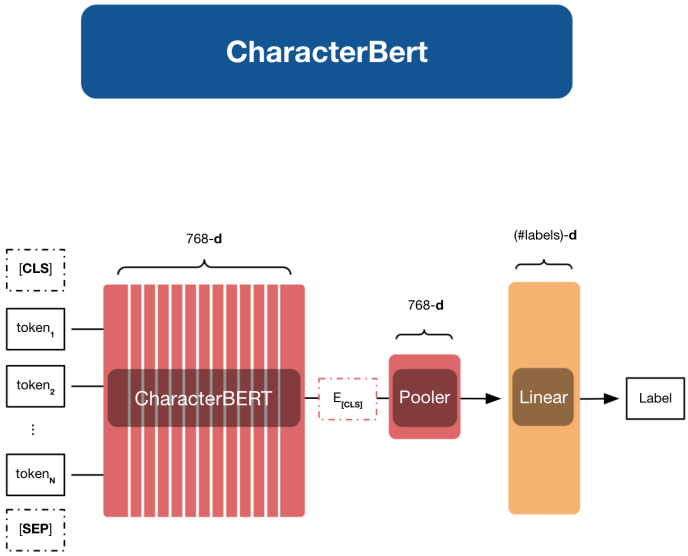

Figure B.8: Knowledge Injection for Classification: CharacterBERT.

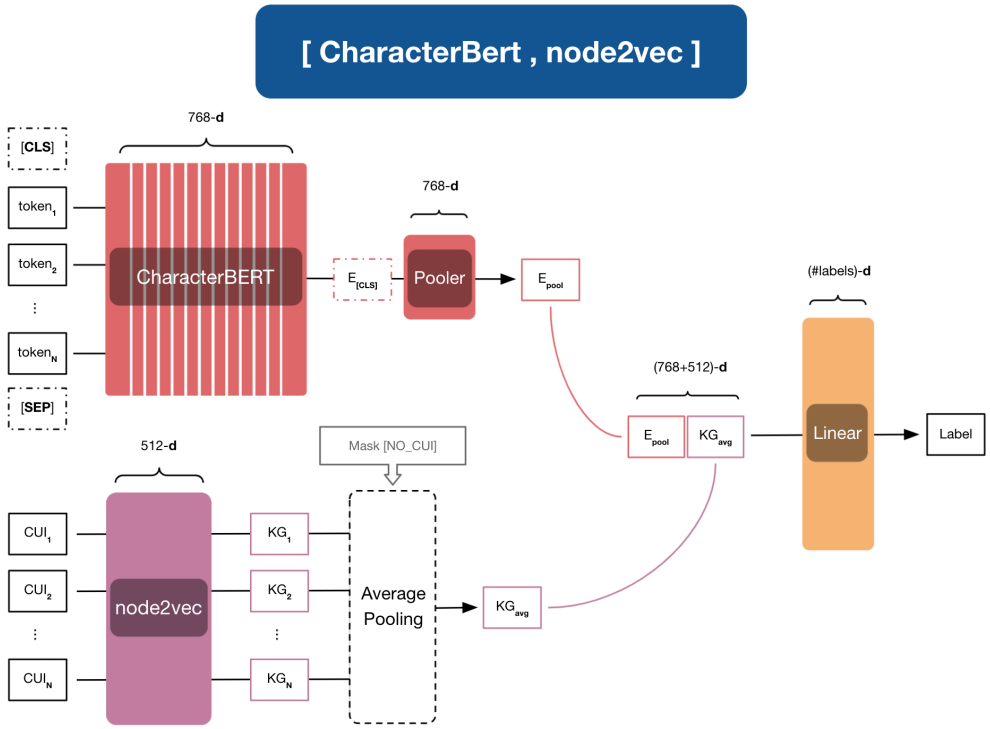

Figure B.9: Knowledge Injection for Classification: [CharacterBERT, node2vec].

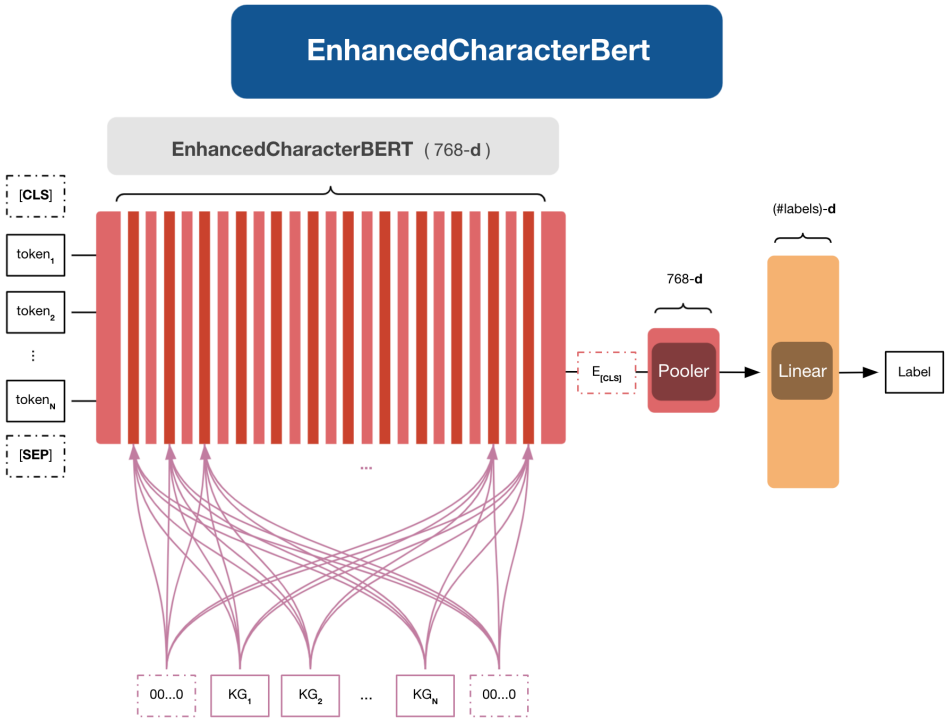

Figure B.10: Knowledge Injection for Classification: EnhancedCharacterBERT.

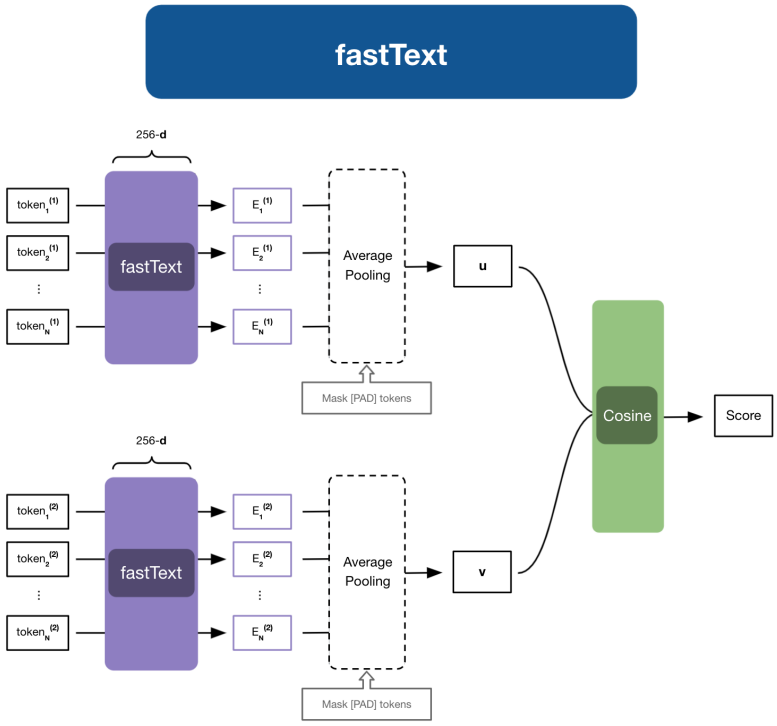

Figure B.11: Knowledge Injection for STS: FASTTEXT.

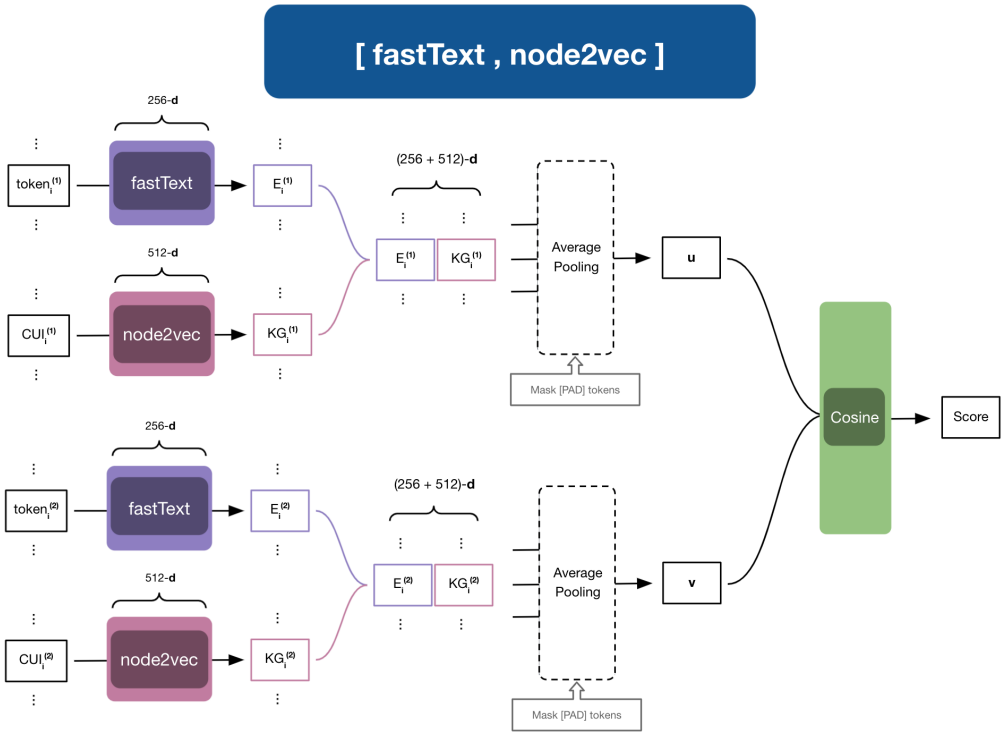

Figure B.12: Knowledge Injection for STS: [fastText, node2vec].

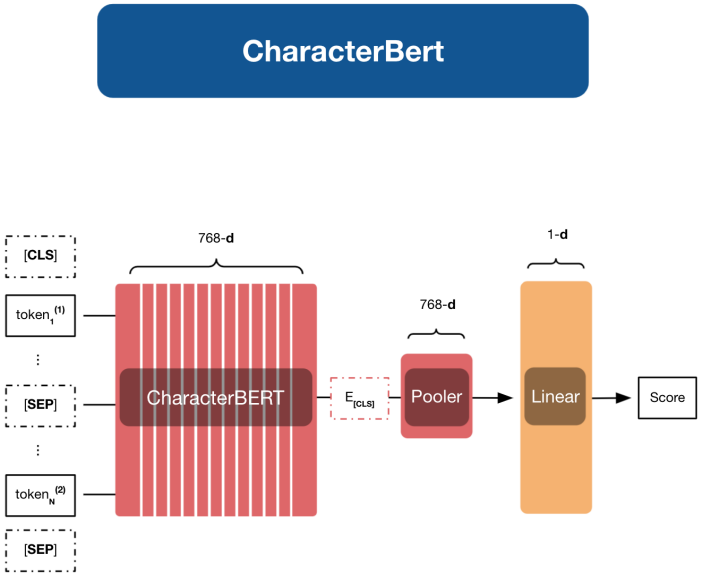

Figure B.13: Knowledge Injection for STS: CharacterBERT.

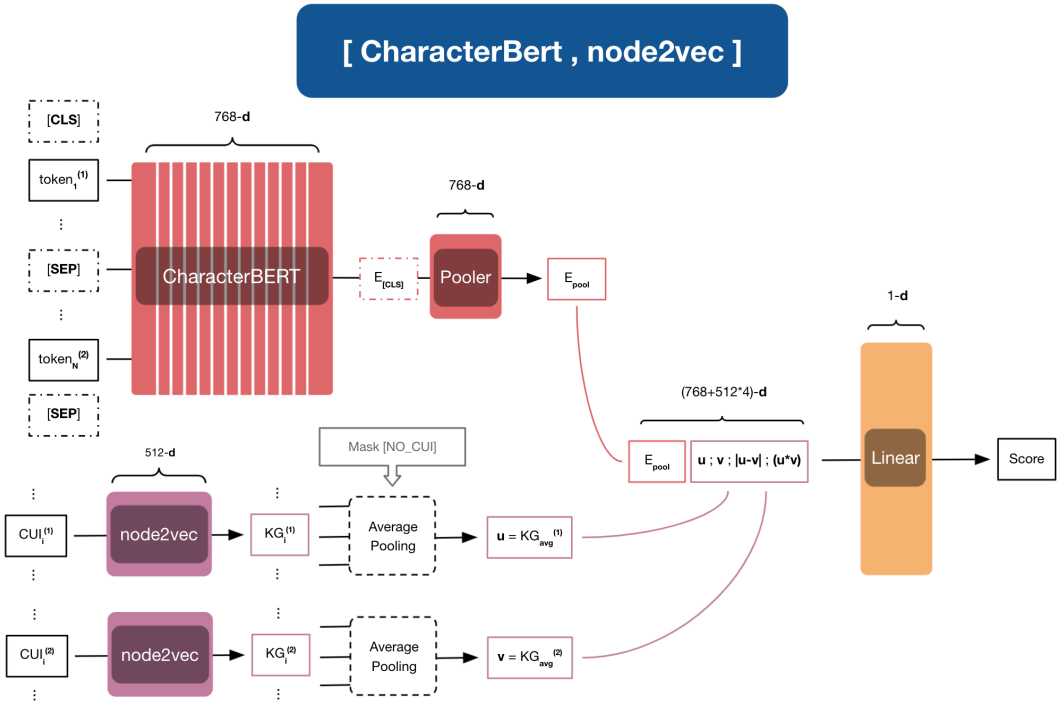

Figure B.14: Knowledge Injection for STS: [CharacterBERT, node2vec].

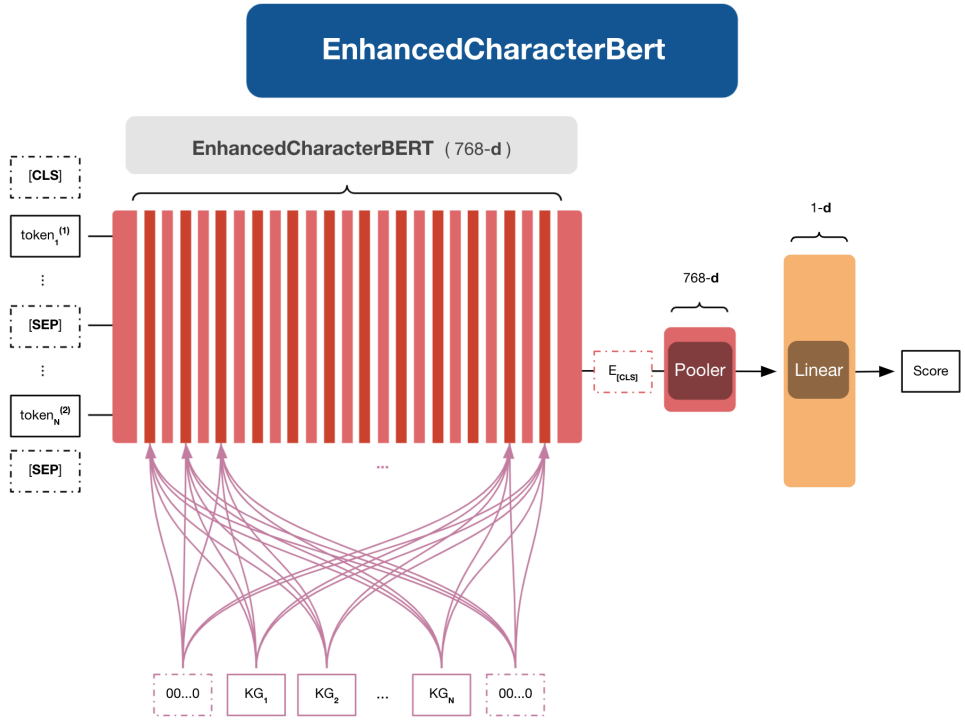

Figure B.15: Knowledge Injection for STS: EnhancedCharacterBERT.

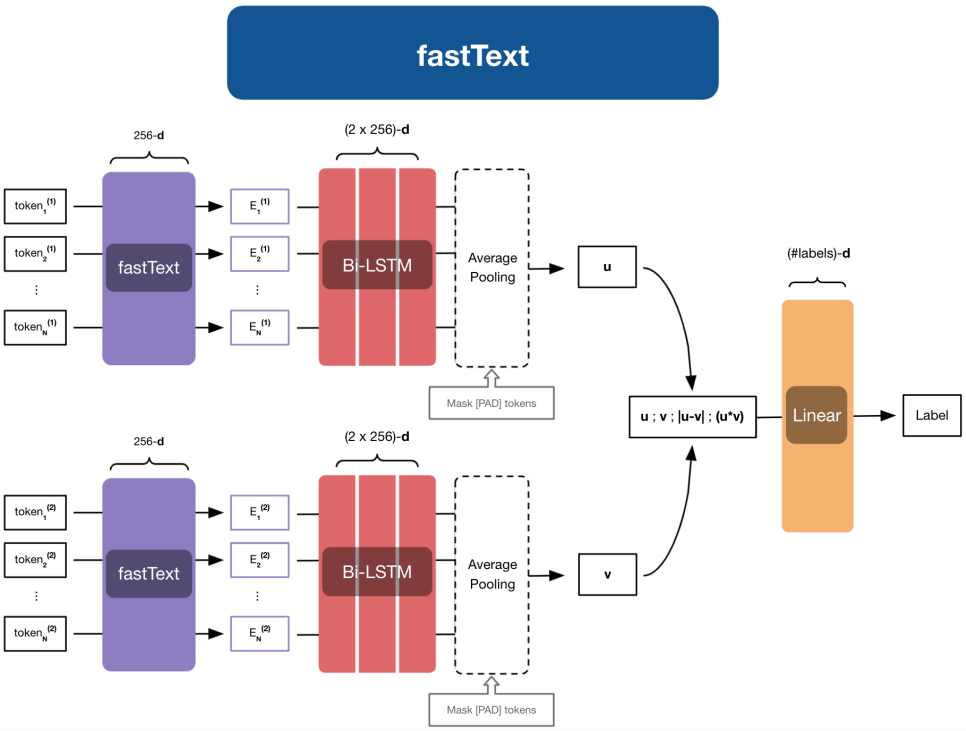

Figure B.16: Knowledge Injection for NLI: FASTTEXT.

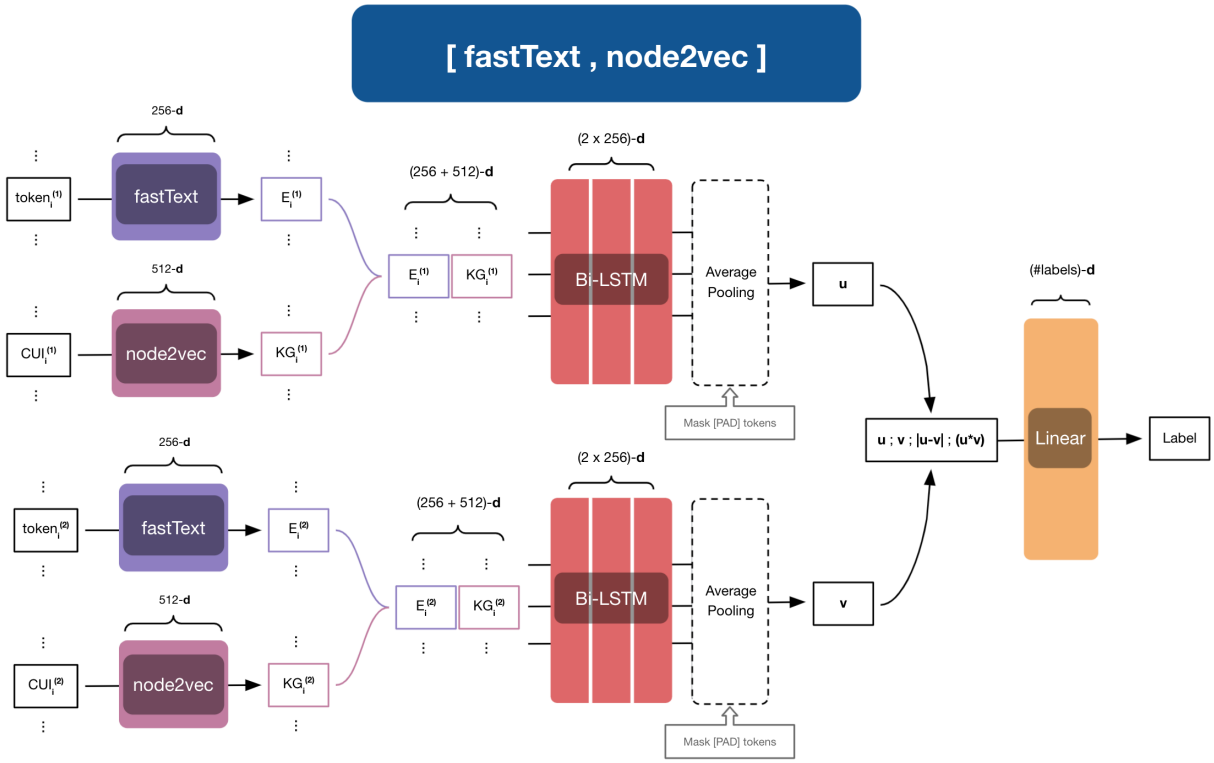

Figure B.17: Knowledge Injection for NLI: [fastText, node2vec].

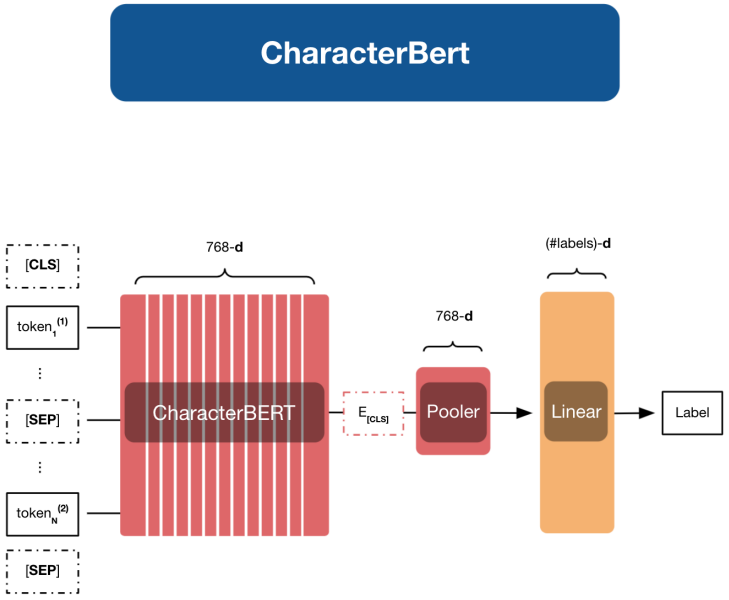

Figure B.18: Knowledge Injection for NLI: CharacterBERT.

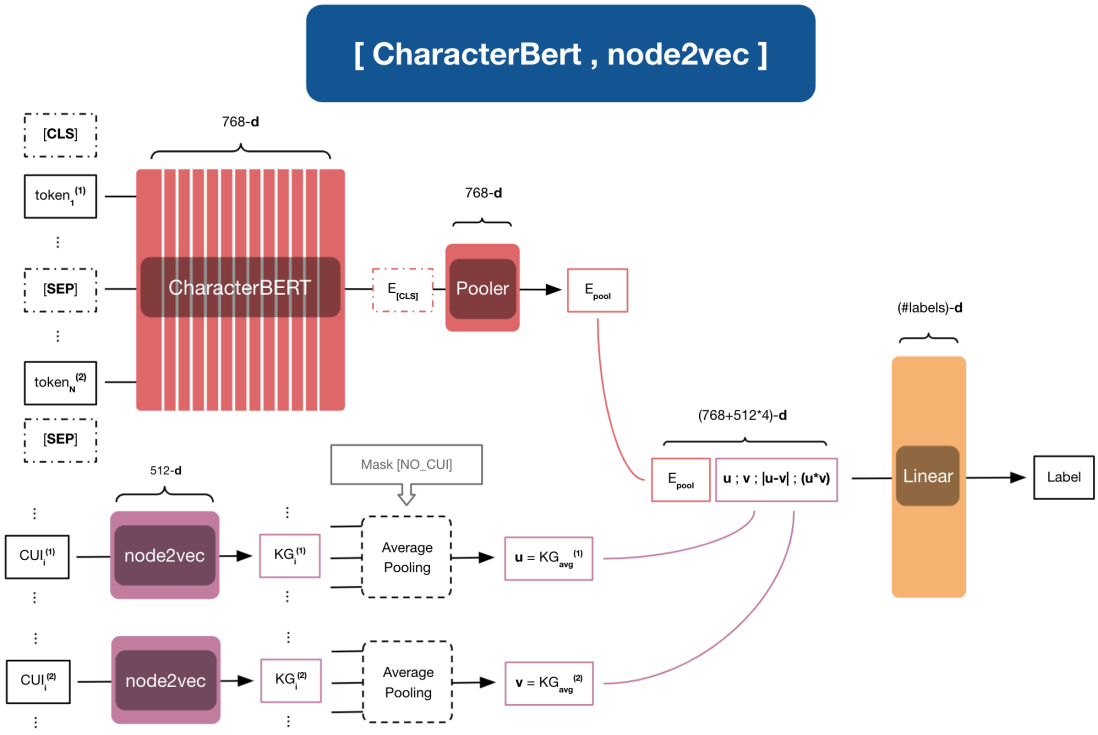

Figure B.19: Knowledge Injection for NLI: [CharacterBERT, node2vec].

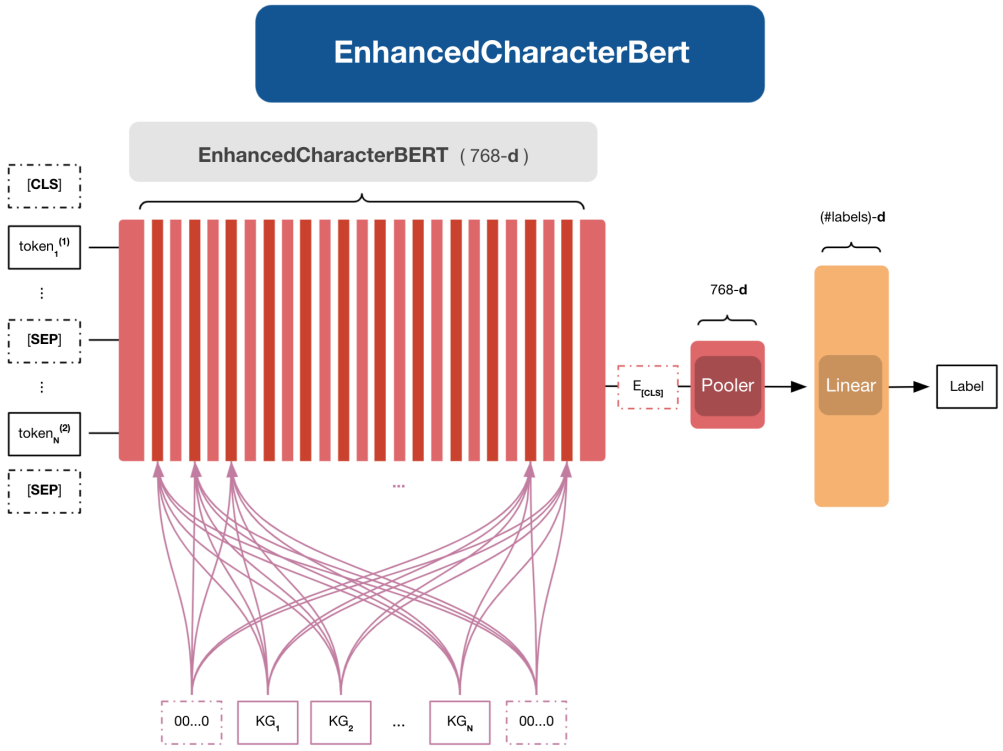

Figure B.20: Knowledge Injection for NLI: EnhancedCharacterBERT.

References

Steven Abney. 2007. *Semisupervised learning for computational linguistics*. CRC Press.

Oshin Agarwal, Heming Ge, Siamak Shakeri, and Rami Al-Rfou. 2021. Knowledge graph based synthetic corpus generation for knowledge-enhanced language model pre-training. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3554–3565, Online. Association for Computational Linguistics.

Emily Alsentzer, John Murphy, William Boag, Wei-Hung Weng, Di Jindi, Tristan Naumann, and Matthew McDermott. 2019. Publicly available clinical BERT embeddings. In *Proceedings of the 2nd Clinical Natural Language Processing Workshop*, pages 72–78, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

Wissam Antoun, Fady Baly, and Hazem Hajj. 2020. AraBERT: Transformer-based model for Arabic language understanding. In *Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection*, pages 9–15, Marseille, France. European Language Resource Association.

Eric Arazo, Diego Ortego, P. Albert, N. O'Connor, and Kevin McGuinness. 2020. Pseudo-labeling and confirmation bias in deep semi-supervised learning. *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8.

Gaurav Arora, Afshin Rahimi, and Timothy Baldwin. 2019. Does an LSTM forget more than a CNN? an empirical study of catastrophic forgetting in NLP. In *Proceedings of the The 17th Annual Workshop of the Australasian Language Technology Association*, pages 77–86, Sydney, Australia. Australasian Language Technology Association.

Muhammad Nabeel Asim, Muhammad Wasim, Muhammad Usman Ghani Khan, Waqar Mahmood, and Hafiza Mahnoor Abbasi. 2018. A survey of ontology learning techniques and applications. *Database*, 2018.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *ArXiv preprint*, abs/1607.06450.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International*

Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings.

Amir Bakarov. 2018. A survey of word embeddings evaluation methods. *ArXiv preprint*, abs/1801.09536.

Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2014. Tailoring continuous word representations for dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 809–815, Baltimore, Maryland. Association for Computational Linguistics.

Richard Bellman. 1966. Dynamic programming. *Science*, 153(3731):34–37.

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A pretrained language model for scientific text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3615–3620, Hong Kong, China. Association for Computational Linguistics.

Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. 2000. A neural probabilistic language model. In *Advances in Neural Information Processing Systems 13, Papers from Neural Information Processing Systems (NIPS) 2000, Denver, CO, USA*, pages 932–938. MIT Press.

Taylor Berg-Kirkpatrick, David Burkett, and Dan Klein. 2012. An empirical investigation of statistical significance in NLP. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 995–1005, Jeju Island, Korea. Association for Computational Linguistics.

Tim Berners-Lee, James Hendler, and Ora Lassila. 2001. The semantic web. *Scientific american*, 284(5):34–43.

Jiang Bian, Bin Gao, and Tie-Yan Liu. 2014. Knowledge-powered deep learning for word embedding. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 132–148. Springer.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2001. Latent dirichlet allocation. In *Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic, NIPS 2001, December 3-8, 2001, Vancouver, British Columbia, Canada]*, pages 601–608. MIT Press.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100.

Olivier Bodenreider. 2004. The Unified Medical Language System (UMLS): integrating biomedical terminology. *Nucleic acids research*, 32(suppl_1):D267–D270.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Danushka Bollegala, Kohei Hayashi, and Ken-ichi Kawarabayashi. 2018. Think globally, embed locally - locally linear meta-embedding of words. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 3970–3976. ijcai.org.

Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 2787–2795.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. 2009. Semi-supervised learning. *IEEE Transactions on Neural Networks*, 20(3):542–542.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *ArXiv preprint*, abs/1312.3005.

Tao Chen, Ruifeng Xu, Yulan He, and Xuan Wang. 2015. Improving distributed representation of word sense via WordNet gloss composition and context clustering. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 15–20, Beijing, China. Association for Computational Linguistics.

Yanqing Chen, Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2013. The expressive power of word embeddings. *ArXiv preprint*, abs/1301.3226.

Jonathan H Clark, Dan Garrette, Iulia Turc, and John Wieting. 2021. CANINE: Pre-training an efficient tokenization-free encoder for language representation. *ArXiv preprint*, abs/2103.06874.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: pre-training text encoders as discriminators rather than generators. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Joshua Coates and Danushka Bollegala. 2018. Frustratingly easy meta-embedding – computing meta-embeddings by averaging source word embeddings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 194–198, New Orleans, Louisiana. Association for Computational Linguistics.

Cody Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter Bailis, Percy Liang, Jure Leskovec, and Matei Zaharia. 2020. Selection via proxy: Efficient data selection for deep learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. In *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*, volume 307 of *ACM International Conference Proceeding Series*, pages 160–167. ACM.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12:2493–2537.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017a. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark. Association for Computational Linguistics.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017b. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, Copenhagen, Denmark. Association for Computational Linguistics.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning*, 20(3):273–297.

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Ziqing Yang, Shijin Wang, and Guoping Hu. 2019. Pre-training with whole word masking for chinese BERT. *ArXiv preprint*, abs/1906.08101.

Andrew M. Dai and Quoc V. Le. 2015. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 3079–3087.

Jifeng Dai, Kaiming He, and Jian Sun. 2016. Instance-aware semantic segmentation via multi-task network cascades. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 3150–3158. IEEE Computer Society.

Dmitry Davidov and Ari Rappoport. 2006. Efficient unsupervised discovery of word categories using symmetric patterns and high frequency words. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 297–304, Sydney, Australia. Association for Computational Linguistics.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. 2009. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, pages 248–255. IEEE Computer Society.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2D knowledge graph embeddings. In *Proceedings of the Thirty-*

Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018, pages 1811–1818. AAAI Press.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Jesse Dodge, Suchin Gururangan, Dallas Card, Roy Schwartz, and Noah A. Smith. 2019. Show your work: Improved reporting of experimental results. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2185–2194, Hong Kong, China. Association for Computational Linguistics.

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. 2020. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *ArXiv preprint*, abs/2002.06305.

Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. 2014. DeCAF: A deep convolutional activation feature for generic visual recognition. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, volume 32 of *JMLR Workshop and Conference Proceedings*, pages 647–655. JMLR.org.

Rotem Dror, Gili Baumer, Segev Shlomov, and Roi Reichart. 2018. The hitchhiker's guide to testing statistical significance in natural language processing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1383–1392, Melbourne, Australia. Association for Computational Linguistics.

Rotem Dror, Segev Shlomov, and Roi Reichart. 2019. Deep dominance - how to properly compare deep neural models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2773–2785, Florence, Italy. Association for Computational Linguistics.

Olive Jean Dunn. 1961. Multiple comparisons among means. *Journal of the American statistical association*, 56(293):52–64.

Maud Ehrmann, Francesco Cecconi, Daniele Vannella, John Philip McCrae, Philipp Cimiano, and Roberto Navigli. 2014. Representing multilingual data as linked data: the case of BabelNet 2.0. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 401–408, Reykjavik, Iceland. European Language Resources Association (ELRA).

Andreas Ekelhart, Stefan Fenz, A Min Tjoa, and Edgar R Weippl. 2007. Security issues for the use of semantic web in e-commerce. In *International Conference on Business Information Systems*, pages 1–13. Springer.

Hicham El Boukkouri. 2020. Ré-entraîner ou entraîner soi-même ? stratégies de pré-entraînement de BERT en domaine médical (re-train or train from scratch ? pre-training strategies for BERT in the medical domain). In *Actes de la 6e conférence conjointe Journées d'Études sur la Parole (JEP, 33e édition), Traitement Automatique des Langues Naturelles (TALN, 27e édition), Rencontre des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues (RÉCITAL, 22e édition). Volume 3 : Rencontre des Étudiants Chercheurs en Informatique pour le TAL*, pages 29–42, Nancy, France. ATALA et AFCP.

Hicham El Boukkouri, Olivier Ferret, Thomas Lavergne, Hiroshi Noji, Pierre Zweigenbaum, and Jun'ichi Tsujii. 2020. CharacterBERT: Reconciling ELMo and BERT for word-level open-vocabulary representations from characters. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6903–6915, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Hicham El Boukkouri, Olivier Ferret, Thomas Lavergne, and Pierre Zweigenbaum. 2019. Embedding strategies for specialized domains: Application to clinical entity recognition. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 295–301, Florence, Italy. Association for Computational Linguistics.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1606–1615, Denver, Colorado. Association for Computational Linguistics.

Manaal Faruqui, Yulia Tsvetkov, Pushpendre Rastogi, and Chris Dyer. 2016. Problems with evaluation of word embeddings using word similarity tasks. In *Proceedings of the 1st Workshop on Evaluating Vector-Space Representations for*

NLP, pages 30–35, Berlin, Germany. Association for Computational Linguistics.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: the concept revisited. In *Proceedings of the Tenth International World Wide Web Conference, WWW 10, Hong Kong, China, May 1-5, 2001*, pages 406–414. ACM.

John R Firth. 1957. A synopsis of linguistic theory, 1930-1955. *Studies in linguistic analysis*.

Robert M French. 1999. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135.

Yoav Freund and Robert E Schapire. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139.

Kunihiko Fukushima and Sei Miyake. 1982. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, pages 267–285. Springer.

Philip Gage. 1994. A new algorithm for data compression. *C Users Journal*, 12(2):23–38.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 758–764, Atlanta, Georgia. Association for Computational Linguistics.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. AllenNLP: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 1–6, Melbourne, Australia. Association for Computational Linguistics.

Goran Glavaš and Ivan Vulić. 2018. Explicit retrofitting of distributional word vectors. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 34–45, Melbourne, Australia. Association for Computational Linguistics.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on*

artificial intelligence and statistics, pages 315–323. JMLR Workshop and Conference Proceedings.

Josu Goikoetxea, Eneko Agirre, and Aitor Soroa. 2016. Single or multiple? combining word representations independently learned from text and wordnet. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pages 2608–2614. AAAI Press.

Josu Goikoetxea, Aitor Soroa, and Eneko Agirre. 2015. Random walks and neural network language models on knowledge bases. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1434–1439, Denver, Colorado. Association for Computational Linguistics.

Aaron Gokaslan and Vanya Cohen. 2019. OpenWebText corpus. `http://Skylion007.github.io/OpenWebTextCorpus`.

Jennifer Golbeck, Gilberto Fragoso, Frank Hartel, Jim Hendler, Jim Oberthaler, and Bijan Parsia. 2003. The national cancer institute's thesaurus and ontology. *Journal of Web Semantics First Look 1_1_4*.

Yoav Goldberg. 2019. Assessing BERT's syntactic abilities. *ArXiv preprint*, abs/1901.05287.

Irving J Good. 1963. Maximum entropy for hypothesis formulation, especially for multidimensional contingency tables. *The Annals of Mathematical Statistics*, 34(3):911–934.

Priya Goyal, Mathilde Caron, Benjamin Lefaudeux, Min Xu, Pengchao Wang, Vivek Pai, Mannat Singh, Vitaliy Liptchinsky, Ishan Misra, Armand Joulin, et al. 2021. Self-supervised pretraining of visual features in the wild. *ArXiv preprint*, abs/2103.01988.

David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2003. English gigaword. *Linguistic Data Consortium, Philadelphia*, 4(1):34.

Arthur Gretton, Alex Smola, Jiayuan Huang, Marcel Schmittfull, Karsten Borgwardt, and Bernhard Schölkopf. 2009. Covariate shift by kernel mean matching. *Dataset shift in machine learning*, 3(4):5.

Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 855–864. ACM.

Thomas R Gruber. 1995. Toward principles for the design of ontologies used for knowledge sharing? *International journal of human-computer studies*, 43(5-6):907–928.

Yu Gu, Robert Tinn, Hao Cheng, Michael Lucas, Naoto Usuyama, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. 2020. Domain-specific language model pretraining for biomedical natural language processing. *ArXiv preprint*, abs/2007.15779.

Jiang Guo, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Revisiting embedding features for simple semi-supervised learning. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 110–120, Doha, Qatar. Association for Computational Linguistics.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online. Association for Computational Linguistics.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. REALM: Retrieval-augmented language model pre-training. *ArXiv preprint*, abs/2002.08909.

Maryam Habibi, Leon Weber, Mariana Neves, David Luis Wiegandt, and Ulf Leser. 2017. Deep learning with word embeddings improves biomedical named entity recognition. *Bioinformatics*, 33(14):i37–i48.

Richard H. R. Hahnloser and H. Sebastian Seung. 2000. Permitted and forbidden sets in symmetric threshold-linear networks. In *Advances in Neural Information Processing Systems 13, Papers from Neural Information Processing Systems (NIPS) 2000, Denver, CO, USA*, pages 217–223. MIT Press.

Richard HR Hahnloser, Rahul Sarpeshkar, Misha A Mahowald, Rodney J Douglas, and H Sebastian Seung. 2000. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, 405(6789):947–951.

Guy Halawi, Gideon Dror, Evgeniy Gabrilovich, and Yehuda Koren. 2012. Large-scale learning of word relatedness with constraints. In *The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, Beijing, China, August 12-16, 2012*, pages 1406–1414. ACM.

Xiaochuang Han and Jacob Eisenstein. 2019. Unsupervised domain adaptation of contextualized embeddings for sequence labeling. In *Proceedings of the*

2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 4238–4248, Hong Kong, China. Association for Computational Linguistics.

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Frederick Hayes-Roth, Donald A Waterman, and Douglas B Lenat. 1983. *Building expert systems*. Addison-Wesley Longman Publishing Co., Inc.

Maryam Hazman, Samhaa R El-Beltagy, and Ahmed Rafea. 2011. A survey of ontology learning approaches. *International Journal of Computer Applications*, 22(9):36–43.

Jingyi He, Kc Tsiolis, Kian Kenyon-Dean, and Jackie Chi Kit Cheung. 2020a. Learning efficient task-specific meta-embeddings with word prisms. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 1229–1241, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Keqing He, Yuanmeng Yan, and Weiran Xu. 2020b. Learning to tag OOV tokens by integrating contextual representation and background knowledge. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 619–624, Online. Association for Computational Linguistics.

Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (GELUs). *ArXiv preprint*, abs/1606.08415.

Felix Hill, Kyunghyun Cho, Sebastien Jean, Coline Devin, and Yoshua Bengio. 2014a. Embedding word similarity with neural machine translation. *ArXiv preprint*, abs/1412.6448.

Felix Hill, KyungHyun Cho, Sebastien Jean, Coline Devin, and Yoshua Bengio. 2014b. Not all neural embeddings are born equal. *ArXiv preprint*, abs/1410.0718.

Felix Hill, Kyunghyun Cho, Anna Korhonen, and Yoshua Bengio. 2016. Learning to understand phrases by embedding the dictionary. *Transactions of the Association for Computational Linguistics*, 4:17–30.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. SimLex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

G. E. Hinton, J. L. McClelland, and D. E. Rumelhart. 1986. *Distributed Representations*, page 77–109. MIT Press, Cambridge, MA, USA.

Tin Kam Ho. 1995. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE.

Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al. 2001. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia. Association for Computational Linguistics.

Eric Huang, Richard Socher, Christopher Manning, and Andrew Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 873–882, Jeju Island, Korea. Association for Computational Linguistics.

Zhiheng Huang, W. Xu, and Kailiang Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *ArXiv preprint*, abs/1508.01991.

Minyoung Huh, Pulkit Agrawal, and Alexei A Efros. 2016. What makes ImageNet good for transfer learning? *ArXiv preprint*, abs/1608.08614.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 448–456. JMLR.org.

Vishal Jain and Mayank Singh. 2013. Ontology based information retrieval in semantic web: A survey. *International Journal of Information Technology and Computer Science (IJITCS)*, 5(10):62.

Abhik Jana and Pawan Goyal. 2018. Can network embedding of distributional thesaurus be combined with word vectors for better representation? In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 463–473, New Orleans, Louisiana. Association for Computational Linguistics.

Sujay Kumar Jauhar, Chris Dyer, and Eduard Hovy. 2015. Ontologically grounded multi-sense representation learning for semantic vector space models. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 683–693, Denver, Colorado. Association for Computational Linguistics.

Pratik Jawanpuria, Satya Dev N T V, Anoop Kunchukuttan, and Bamdev Mishra. 2020. Learning geometric word meta-embeddings. In *Proceedings of the 5th Workshop on Representation Learning for NLP*, pages 39–44, Online. Association for Computational Linguistics.

Edwin T Jaynes. 1957. Information theory and statistical mechanics. *Physical review*, 106(4):620.

Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S. Yu. 2021. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–21.

Jing Jiang. 2008. A literature survey on domain adaptation of statistical classifiers. *URL: http://sifaka. cs. uiuc. edu/jiang4/domainadaptation/survey*, 3(1-12):3.

Min Jiang, Todd Sanger, and Xiong Liu. 2019. Combining contextualized embeddings and prior knowledge for clinical named entity recognition: evaluation study. *JMIR medical informatics*, 7(4):e14850.

Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-Wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. 2016. MIMIC-III, a freely accessible critical care database. *Scientific data*, 3(1):1–9.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *ArXiv preprint*, abs/1602.02410.

Andrej Karpathy and Fei-Fei Li. 2015. Deep visual-semantic alignments for generating image descriptions. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 3128–3137. IEEE Computer Society.

Douwe Kiela, Alexis Conneau, Allan Jabri, and Maximilian Nickel. 2018a. Learning visually grounded sentence representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 408–418, New Orleans, Louisiana. Association for Computational Linguistics.

Douwe Kiela, Changhan Wang, and Kyunghyun Cho. 2018b. Dynamic meta-embeddings for improved sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1466–1477, Brussels, Belgium. Association for Computational Linguistics.

Joo-Kyung Kim, Marie-Catherine de Marneffe, and Eric Fosler-Lussier. 2016a. Adjusting word embeddings with semantic intensity orders. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 62–69, Berlin, Germany. Association for Computational Linguistics.

Yoon Kim. 2014a. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Yoon Kim. 2014b. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Yoon Kim, Yacine Jernite, David A. Sontag, and Alexander M. Rush. 2016b. Character-aware neural language models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pages 2741–2749. AAAI Press.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Barbara Ann Kipfer. 1993. *Roget's 21st century thesaurus in dictionary form: the essential reference for home, school, or office*. Laurel.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526.

Dennis H Klatt. 1987. Review of text-to-speech conversion for english. *The Journal of the Acoustical Society of America*, 82(3):737–793.

Yuri Kuratov and Mikhail Arkhipov. 2019. Adaptation of deep bidirectional multilingual transformers for russian language. *ArXiv preprint*, abs/1905.07213.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001), Williams College, Williamstown, MA, USA, June 28 - July 1, 2001*, pages 282–289. Morgan Kaufmann.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California. Association for Computational Linguistics.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Thomas K Landauer and Susan T Dumais. 1997. A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*, 104(2):211.

Hang Le, Loïc Vial, Jibril Frej, Vincent Segonne, Maximin Coavoux, Benjamin Lecouteux, Alexandre Allauzen, Benoit Crabbé, Laurent Besacier, and Didier Schwab. 2020. FlauBERT: Unsupervised language model pre-training for French. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 2479–2490, Marseille, France. European Language Resources Association.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. 1989. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551.

Alessandro Lenci, Simonetta Montemagni, Vito Pirrelli, and Giulia Venturi. 2007. NLP-based ontology learning from legal texts. a case study. *LOAIT*, 321:113–129.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

Jiao Li, Yueping Sun, Robin J Johnson, Daniela Sciaky, Chih-Hsuan Wei, Robert Leaman, Allan Peter Davis, Carolyn J Mattingly, Thomas C Wiegers, and Zhiyong Lu. 2016. BioCreative v CDR task corpus: a resource for chemical disease relation extraction. *Database*, 2016.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, pages 2181–2187. AAAI Press.

Yongjie Lin, Yi Chern Tan, and Robert Frank. 2019. Open sesame: Getting inside BERT's linguistic knowledge. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 241–253, Florence, Italy. Association for Computational Linguistics.

Quan Liu, Hui Jiang, Si Wei, Zhen-Hua Ling, and Yu Hu. 2015. Learning semantic word embeddings based on ordinal knowledge constraints. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1501–1511, Beijing, China. Association for Computational Linguistics.

Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. 2020. K-BERT: enabling language representation with knowledge graph. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 2901–2908. AAAI Press.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *ArXiv preprint*, abs/1907.11692.

Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet. 2018. Are GANs created equal? A large-scale study. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 698–707.

Hans Peter Luhn. 1957. A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of research and development*, 1(4):309–317.

Yong Luo, Jian Tang, Jun Yan, Chao Xu, and Zheng Chen. 2014. Pre-trained multi-view word embedding using two-side neural network. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada*, pages 1982–1988. AAAI Press.

Minh-Thang Luong and Christopher D. Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1054–1063, Berlin, Germany. Association for Computational Linguistics.

Wentao Ma, Yiming Cui, Chenglei Si, Ting Liu, Shijin Wang, and Guoping Hu. 2020. CharBERT: Character-aware pre-trained language model. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 39–50, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Xiaofei Ma, Peng Xu, Zhiguo Wang, Ramesh Nallapati, and Bing Xiang. 2019. Domain adaptation with BERT-based domain classification and data selection. In *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*, pages 76–83, Hong Kong, China. Association for Computational Linguistics.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.

Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al. 2013. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3. Citeseer.

Christopher D Manning, Kevin Clark, John Hewitt, Urvashi Khandelwal, and Omer Levy. 2020. Emergent linguistic structure in artificial neural networks trained by self-supervision. *Proceedings of the National Academy of Sciences*, 117(48):30046–30054.

Anna Margolis. 2011. A literature review of domain adaptation with unlabeled data. *Tec. Report*, pages 1–42.

Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suárez, Yoann Dupont, Laurent Romary, Éric de la Clergerie, Djamé Seddah, and Benoît Sagot. 2020. CamemBERT: a tasty French language model. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7203–7219, Online. Association for Computational Linguistics.

Michael L. Mauldin. 1984. Semantic rule based text generation. In *10th International Conference on Computational Linguistics and 22nd Annual Meeting of the Association for Computational Linguistics*, pages 376–380, Stanford, California, USA. Association for Computational Linguistics.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 6294–6305.

Michael McCloskey and Neal J Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier.

N Madurai Meenachi and M Sai Baba. 2012. A survey on usage of ontology in different domains. *International Journal of Applied Information Systems*, 4(2):46–55.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional LSTM. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 51–61, Berlin, Germany. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *ArXiv preprint*, abs/1301.3781.

George A Miller. 1998. *WordNet: An electronic lexical database*. MIT press.

Marvin L Minsky and Seymour A Papert. 1988. Perceptrons: expanded edition.

Andriy Mnih and Geoffrey E. Hinton. 2008. A scalable hierarchical distributed language model. In *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 8-11, 2008*, pages 1081–1088. Curran Associates, Inc.

Andriy Mnih and Koray Kavukcuoglu. 2013. Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 2265–2273.

Robert C. Moore and William Lewis. 2010. Intelligent selection of language model training data. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 220–224, Uppsala, Sweden. Association for Computational Linguistics.

Lili Mou, Zhao Meng, Rui Yan, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2016. How transferable are neural networks in NLP applications? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 479–489, Austin, Texas. Association for Computational Linguistics.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Lina M. Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting word vectors to linguistic constraints. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 142–148, San Diego, California. Association for Computational Linguistics.

Nikola Mrkšić, Ivan Vulić, Diarmuid Ó Séaghdha, Ira Leviant, Roi Reichart, Milica Gašić, Anna Korhonen, and Steve Young. 2017. Semantic specialization of distributional word vector spaces using monolingual and cross-lingual constraints. *Transactions of the Association for Computational Linguistics*, 5:309–324.

Roberto Navigli and Simone Paolo Ponzetto. 2012. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial intelligence*, 193:217–250.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2014. Efficient non-parametric estimation of multiple embeddings per word in vector space. In *Proceedings of the 2014 Conference on Empirical Methods*

in Natural Language Processing (EMNLP), pages 1059–1069, Doha, Qatar. Association for Computational Linguistics.

Kim Anh Nguyen, Sabine Schulte im Walde, and Ngoc Thang Vu. 2016. Integrating distributional lexical contrast into word embeddings for antonym-synonym distinction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 454–459, Berlin, Germany. Association for Computational Linguistics.

Borys Omelayenko. 2001. Learning of ontologies from the web: the analysis of existent approaches. In *WebDyn@ ICDT*, pages 16–25. Citeseer.

Masataka Ono, Makoto Miwa, and Yutaka Sasaki. 2015. Word embedding-based antonym detection using thesauri and distributional information. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 984–989, Denver, Colorado. Association for Computational Linguistics.

Maxime Oquab, Léon Bottou, Ivan Laptev, and Josef Sivic. 2014. Learning and transferring mid-level image representations using convolutional neural networks. In *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*, pages 1717–1724. IEEE Computer Society.

Massimo Paolucci, Katia Sycara, Takuya Nishimura, and Naveen Srinivasan. 2003. Toward a semantic web e-commerce. In *Proc. of 6th Int. Conf. on Business Information Systems (BIS'2003)*.

Yifan Peng, Shankai Yan, and Zhiyong Lu. 2019. Transfer learning in biomedical natural language processing: An evaluation of BERT and ELMo on ten benchmarking datasets. In *Proceedings of the 18th BioNLP Workshop and Shared Task*, pages 58–65, Florence, Italy. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Asunción Gómez Pérez and V Richard Benjamins. 1999. Overview of knowledge sharing and reuse components: Ontologies and problem-solving methods. In *Proceedings of the IJCAI-99 workshop on Ontologies and Problem-Solving methods (KRR5), Stockholm, Sweden*, pages 1–15.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: online learning of social representations. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, pages 701–710. ACM.

Matthew E. Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1756–1765, Vancouver, Canada. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018a. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Robert Logan, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. 2019. Knowledge enhanced contextual word representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 43–54, Hong Kong, China. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Luke Zettlemoyer, and Wen-tau Yih. 2018b. Dissecting contextual word embeddings: Architecture and representation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509, Brussels, Belgium. Association for Computational Linguistics.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Jason Phang, Iacer Calixto, Phu Mon Htut, Yada Pruksachatkun, Haokun Liu, Clara Vania, Katharina Kann, and Samuel R. Bowman. 2020. English intermediate-task training improves zero-shot cross-lingual transfer too. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference*

on Natural Language Processing, pages 557–575, Suzhou, China. Association for Computational Linguistics.

Jason Phang, Thibault Févry, and Samuel R Bowman. 2018. Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks. *ArXiv preprint*, abs/1811.01088.

Barbara Plank. 2016. What to do about non-standard (or non-canonical) language in NLP. *ArXiv preprint*, abs/1608.07836.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 412–418, Berlin, Germany. Association for Computational Linguistics.

Barbara Plank and Gertjan van Noord. 2011. Effective measures of domain similarity for parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1566–1576, Portland, Oregon, USA. Association for Computational Linguistics.

Danish Pruthi, Bhuwan Dhingra, and Zachary C. Lipton. 2019. Combating adversarial misspellings with robust word recognition. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5582–5591, Florence, Italy. Association for Computational Linguistics.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. Technical report.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *ArXiv preprint*, abs/1910.10683.

Alan Ramponi and Barbara Plank. 2020. Neural unsupervised domain adaptation in NLP—A survey. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6838–6855, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Kumar Ravi and Vadlamani Ravi. 2015. A survey on opinion mining and sentiment analysis: tasks, approaches and applications. *Knowledge-based systems*, 89:14–46.

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. 2017. Learning multiple visual domains with residual adapters. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 506–516.

Alan Rector, Stefan Schulz, Jean Marie Rodrigues, Christopher G Chute, and Harold Solbrig. 2019. On beyond gruber: "ontologies" in today's biomedical information systems and the limits of OWL. *Journal of Biomedical Informatics: X*, 2:100002.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of LSTM-networks for sequence tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 338–348, Copenhagen, Denmark. Association for Computational Linguistics.

Robert Remus. 2012. Domain adaptation using domain similarity-and domain complexity-based instance selection for cross-domain sentiment analysis. In *2012 IEEE 12th international conference on data mining workshops*, pages 717–723. IEEE.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5418–5426, Online. Association for Computational Linguistics.

Kirk Roberts. 2016. Assessing the corpus size vs. similarity trade-off for word embeddings in clinical NLP. In *Proceedings of the Clinical Natural Language Processing Workshop (ClinicalNLP)*, pages 54–63, Osaka, Japan. The COLING 2016 Organizing Committee.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in BERTology: What we know about how BERT works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

Alexey Romanov and Chaitanya Shivade. 2018. Lessons from natural language inference in the clinical domain. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1586–1596, Brussels, Belgium. Association for Computational Linguistics.

Sebastian Ruder. 2019. *Neural transfer learning for natural language processing*. Ph.D. thesis, NUI Galway.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1986. Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. 2015. ImageNet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252.

Ruslan Salakhutdinov. 2014. Deep learning. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, page 1973. ACM.

Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. 2018. How does batch normalization help optimization? In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 2488–2498.

Jürgen Schmidhuber. 2015. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117.

Tobias Schnabel, Igor Labutov, David Mimno, and Thorsten Joachims. 2015. Evaluation methods for unsupervised word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 298–307, Lisbon, Portugal. Association for Computational Linguistics.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. FaceNet: A unified embedding for face recognition and clustering. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 815–823. IEEE Computer Society.

Roy Schwartz, Roi Reichart, and Ari Rappoport. 2015. Symmetric pattern based word embeddings for improved word similarity prediction. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 258–267, Beijing, China. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Sapan Shah, Sreedhar Reddy, and Pushpak Bhattacharyya. 2020. A retrofitting model for incorporating semantic relations into word embeddings. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 1292–1298, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Ling Shao, Fan Zhu, and Xuelong Li. 2014. Transfer learning for visual categorization: A survey. *IEEE transactions on neural networks and learning systems*, 26(5):1019–1034.

Weijia Shi, Muhao Chen, Pei Zhou, and Kai-Wei Chang. 2019. Retrofitting contextualized word embeddings with paraphrases. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1198–1203, Hong Kong, China. Association for Computational Linguistics.

Yuqi Si, Jingqi Wang, Hua Xu, and Kirk Roberts. 2019. Enhancing clinical concept extraction with contextual embeddings. *Journal of the American Medical Informatics Association*, 26(11):1297–1304.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Gizem Soğancıoğlu, Hakime Öztürk, and Arzucan Özgür. 2017. BIOSSES: a semantic sentence similarity estimation system for the biomedical domain. *Bioinformatics*, 33(14):i49–i58.

Kent A Spackman, Keith E Campbell, and Roger A Côté. 1997. SNOMED RT: a reference terminology for health care. In *Proceedings of the AMIA annual fall symposium*, page 640. American Medical Informatics Association.

Karen Spärck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *ArXiv preprint*, abs/1505.00387.

Milan Straka, Jakub Náplava, Jana Straková, and David Samuel. 2021. RobeCzech: Czech RoBERTa, a monolingual contextualized language representation model. *ArXiv preprint*, abs/2105.11314.

Student. 1908. The probable error of a mean. *Biometrika*, pages 1–25.

Lichao Sun, Kazuma Hashimoto, Wenpeng Yin, Akari Asai, Jia Li, Philip Yu, and Caiming Xiong. 2020. Adv-BERT: BERT is not robust on misspellings! generating nature adversarial samples on BERT. *ArXiv preprint*, abs/2003.04985.

Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. 2018. A survey on deep transfer learning. In *International conference on artificial neural networks*, pages 270–279. Springer.

Yi Tay, Vinh Q Tran, Sebastian Ruder, Jai Gupta, Hyung Won Chung, Dara Bahri, Zhen Qin, Simon Baumgartner, Cong Yu, and Donald Metzler. 2021. Charformer: Fast character transformers via gradient-based subword tokenization. *ArXiv preprint*, abs/2106.12672.

Wilson L Taylor. 1953. "cloze procedure": A new tool for measuring readability. *Journalism quarterly*, 30(4):415–433.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R. Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R. Bowman, Dipanjan Das, and Ellie Pavlick. 2019. What do you learn from context? probing for sentence structure in contextualized word representations. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Steffen Thoma, Achim Rettinger, and F. Both. 2017. Knowledge fusion via embeddings from text, knowledge graphs, and images. *ArXiv preprint*, abs/1704.06084.

Andrew Trask, Phil Michalak, and John Liu. 2015. sense2vec-a fast and accurate method for word sense disambiguation in neural word embeddings. *ArXiv preprint*, abs/1511.06388.

Yuta Tsuboi. 2014. Neural networks leverage corpus-wide information for part-of-speech tagging. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 938–950, Doha, Qatar. Association for Computational Linguistics.

Yulia Tsvetkov, Manaal Faruqui, Wang Ling, Guillaume Lample, and Chris Dyer. 2015. Evaluation of word vector representations by subspace alignment. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2049–2054, Lisbon, Portugal. Association for Computational Linguistics.

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394, Uppsala, Sweden. Association for Computational Linguistics.

Peter D Turney, Michael L Littman, Jeffrey Bigham, and Victor Shnayder. 2003. Combining independent modules to solve multiple-choice synonym and analogy problems. *arXiv preprint cs/0309035*.

Özlem Uzuner, Brett R South, Shuying Shen, and Scott L DuVall. 2011. 2010 i2b2/VA challenge on concepts, assertions, and relations in clinical text. *Journal of the American Medical Informatics Association*, 18(5):552–556.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Ivan Vulić, Goran Glavaš, Nikola Mrkšić, and Anna Korhonen. 2018. Post-specialisation: Retrofitting vectors of words unseen in lexical resources. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 516–527, New Orleans, Louisiana. Association for Computational Linguistics.

Dong Wang and Thomas Fang Zheng. 2015. Transfer learning for speech and language processing. In *2015 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, pages 1225–1237. IEEE.

Jiang Wang, Yang Song, Thomas Leung, Chuck Rosenberg, Jingbin Wang, James Philbin, Bo Chen, and Ying Wu. 2014. Learning fine-grained image similarity

with deep ranking. In *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*, pages 1386–1393. IEEE Computer Society.

Qi Wang, Yangming Zhou, Tong Ruan, Daqi Gao, Yuhang Xia, and Ping He. 2019. Incorporating dictionaries into deep neural networks for the chinese clinical named entity recognition. *Journal of biomedical informatics*, 92:103133.

Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuanjing Huang, Jianshu Ji, Guihong Cao, Daxin Jiang, and Ming Zhou. 2021. K-Adapter: Infusing Knowledge into Pre-Trained Models with Adapters. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1405–1418, Online. Association for Computational Linguistics.

Yanshan Wang, Naveed Afzal, Sunyang Fu, Liwei Wang, Feichen Shen, Majid Rastegar-Mojarad, and Hongfang Liu. 2020. MedSTS: a resource for clinical semantic textual similarity. *Language Resources and Evaluation*, 54(1):57–72.

Yanshan Wang, Naveed Afzal, Sijia Liu, Majid Rastegar-Mojarad, Liwei Wang, Feichen Shen, Sunyang Fu, and Hongfang Liu. 2018. Overview of the BioCreative/OHNLP challenge 2018 task 2: clinical semantic textual similarity. *Proceedings of the BioCreative/OHNLP Challenge*, 2018.

Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. 2016. A survey of transfer learning. *Journal of Big data*, 3(1):1–40.

Paul J Werbos. 1990. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. From paraphrase database to compositional paraphrase model and back. *Transactions of the Association for Computational Linguistics*, 3:345–358.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Towards universal paraphrastic sentence embeddings. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Stephen Wu, Kirk Roberts, Surabhi Datta, Jingcheng Du, Zongcheng Ji, Yuqi Si, Sarvesh Soni, Qiong Wang, Qiang Wei, Yang Xiang, et al. 2020. Deep learning in clinical natural language processing: a methodical review. *Journal of the American Medical Informatics Association*, 27(3):457–470.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *ArXiv preprint*, abs/1609.08144.

Yonghui Wu, Xi Yang, Jiang Bian, Yi Guo, Hua Xu, and William Hogan. 2018. Combine factual medical knowledge and distributed word representation to improve clinical named entity recognition. In *AMIA Annual Symposium Proceedings*, volume 2018, page 1110. American Medical Informatics Association.

Caiming Xiong, Victor Zhong, and Richard Socher. 2018. DCN+: mixed objective and deep residual coattention for question answering. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Chang Xu, Yalong Bai, Jiang Bian, Bin Gao, Gang Wang, Xiaoguang Liu, and Tie-Yan Liu. 2014. RC-NET: A general framework for incorporating knowledge into word representations. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China, November 3-7, 2014*, pages 1219–1228. ACM.

Jingjing Xu, Hangfeng He, Xu Sun, Xuancheng Ren, and Sujian Li. 2018. Cross-domain and semisupervised named entity recognition in chinese social media: A unified model. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(11):2142–2152.

Qiantong Xu, T. Likhomanenko, Jacob Kahn, Awni Y. Hannun, Gabriel Synnaeve, and Ronan Collobert. 2020. Iterative pseudo-labeling for speech recognition. *ArXiv preprint*, abs/2005.09267.

Yi Yang, Mark Christopher Siy Uy, and Allen Huang. 2020. FinBERT: A pretrained language model for financial communications. *ArXiv preprint*, abs/2006.08097.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. XLNet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 5754–5764.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. KG-BERT: BERT for knowledge graph completion. *ArXiv preprint*, abs/1909.03193.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196, Cambridge, Massachusetts, USA. Association for Computational Linguistics.

Keiji Yasuda, Ruiqiang Zhang, Hirofumi Yamamoto, and Eiichiro Sumita. 2008. Method of selecting training data to build a compact and efficient translation model. In *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-II*.

Wenpeng Yin and Hinrich Schütze. 2015. Multichannel variable-size convolution for sentence classification. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 204–214, Beijing, China. Association for Computational Linguistics.

Wenpeng Yin and Hinrich Schütze. 2016. Learning word meta-embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1351–1360, Berlin, Germany. Association for Computational Linguistics.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3320–3328.

Yang You, Jing Li, Sashank J. Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. 2020. Large batch optimization for deep learning: Training BERT in 76 minutes. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Mo Yu and Mark Dredze. 2014. Improving lexical embeddings with semantic knowledge. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 545–550, Baltimore, Maryland. Association for Computational Linguistics.

Ye Zhang, Stephen Roller, and Byron C. Wallace. 2016. MGNC-CNN: A simple approach to exploiting multiple word embeddings for sentence classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1522–1527, San Diego, California. Association for Computational Linguistics.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: Enhanced language representation with informative entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1441–1451, Florence, Italy. Association for Computational Linguistics.

Zhuosheng Zhang, Yuwei Wu, Hai Zhao, Zuchao Li, Shuailiang Zhang, Xi Zhou, and Xiang Zhou. 2020. Semantics-aware BERT for language understanding. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 9628–9635. AAAI Press.

Zhi-Hua Zhou and Ming Li. 2005. Tri-training: Exploiting unlabeled data using three classifiers. *IEEE Transactions on knowledge and Data Engineering*, 17(11):1529–1541.

Haotian Zhu, Denise Mak, Jesse Gioannini, and Fei Xia. 2020. NLPStatTest: A toolkit for comparing NLP system performance. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 40–46, Suzhou, China. Association for Computational Linguistics.

Henghui Zhu, Ioannis Ch. Paschalidis, and Amir Tahmasebi. 2018. Clinical concept extraction with contextual word embedding. volume abs/1810.10566.

Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 19–27. IEEE Computer Society.

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre: Adaptation au domaine de plongements lexicaux via l'exploitation de corpus et de bases de connaissances spécialisés

Mots clés: Plongements lexicaux, Domaine spécialisé, Domaine médical, Adaptation au domaine, Traitement automatique des langues, Base de connaissances

Résumé: Il existe, à la base de la plupart des systèmes de TAL, des représentations numériques appelées « plongements lexicaux » qui permettent à la machine de traiter, d'interagir avec et, dans une certaine mesure, de comprendre le langage humain. Ces plongements lexicaux nécessitent une quantité importante de textes afin d'être entraînés correctement, ce qui conduit souvent les praticiens du TAL à collecter et fusionner des textes provenant de sources multiples, mélangeant souvent différents styles et domaines (par exemple, des encyclopédies, des articles de presse, des articles scientifiques, etc.). Ces corpus dits du « domaine général » sont aujourd'hui la base sur laquelle s'entraînent la plupart des plongements lexicaux, limitant fortement leur utilisation dans des domaines plus spécifiques. En effet, les « domaines spécialisés » comme le domaine médical manifestent généralement assez de spécificités lexicales, sémantiques et stylistiques (par exemple, l'utilisation d'acronymes et de termes techniques) pour que les plongements lexicaux généraux ne soient pas en mesure de les représenter efficacement. Dans le cadre de cette thèse, nous explorons comment différents types de ressources peuvent être exploités afin soit d'entraîner de nouveaux plongements spécialisés, soit de spécialiser davantage des représentations préexistantes.

Plus précisément, nous étudions d'abord comment des corpus de textes peuvent être utilisés à cette fin. En particulier, nous montrons que la taille du corpus ainsi que son degré de similarité au domaine d'intérêt jouent un rôle important dans ce processus puis proposons un moyen de tirer parti d'un petit corpus du domaine cible afin d'obtenir de meilleurs résultats dans des contextes à faibles ressources. Ensuite, nous abordons le cas des modèles de type BERT et observons que les vocabulaires généraux de ces modèles conviennent mal aux domaines spécialisés. Cependant, nous montrons des résultats indiquant que des modèles formés à l'aide de tels vocabulaires peuvent néanmoins être comparables à des systèmes entièrement spécialisés et utilisant des vocabulaires du domaine du domaine, ce qui nous amène à la conclusion que le ré-entraînement de modèles du domaine général est une approche tout à fait efficace pour construire des systèmes spécialisés. Nous proposons également CharacterBERT, une variante de BERT capable de produire des représentations de mots entiers en vocabulaire ouvert via la consultation des caractères de ces mots. Nous montrons des résultats indiquant que cette architecture conduit à une amélioration des performances dans le domaine médical tout en étant plus robuste aux fautes d'orthographe.

Enfin, nous étudions comment des ressources externes sous forme de bases de connaissances et ontologies du domaine peuvent être exploitées pour spécialiser des représentations de mots préexistantes. Dans ce cadre, nous proposons une approche simple qui consiste à construire des représentations denses de bases de connaissances puis à combiner ces "vecteurs de connaissances" avec les plongements lexicaux cibles. Nous généralisons cette approche et proposons également des Modules d'Injection de Connaissances, de petites couches neuronales permettant l'intégration de représentations de connaissances externes au sein des couches cachées de modèles à base de Transformers. Globalement, nous montrons que ces approches peuvent conduire à de meilleurs résultats, cependant, nous avons l'intuition que ces performances finales dépendent en fin de compte de la disponibilité de connaissances pertinentes pour la tâche cible au sein des bases de connaissances considérées.

Dans l'ensemble, notre travail montre que les corpus et bases de connaissances du domaine peuvent être utilisés pour construire de meilleurs plongements lexicaux en domaine spécialisé. Enfin, afin de faciliter les recherches futures sur des sujets similaires, nous publions notre code et partageons autant que possible nos modèles pré-entraînés.

Title: Domain adaptation of word embeddings through the exploitation of in-domain corpora and knowledge bases

Keywords: Word Embeddings, Specialized Domain, Medical Domain, Domain Adaptation, Natural Language Processing, Knowledge Base

Abstract: There are, at the basis of most NLP systems, numerical representations that enable the machine to process, interact with and—to some extent—understand human language. These "word embeddings" come in different flavours but can be generally categorised into two distinct groups: on one hand, static embeddings that learn and assign a single definitive representation to each word; and on the other, contextual embeddings that instead learn to generate word representations on the fly, according to a current context. In both cases, training these models requires a large amount of texts. This often leads NLP practitioners to compile and merge texts from multiple sources, often mixing different styles and domains (e.g. encyclopaedias, news articles, scientific articles, etc.) in order to produce corpora that are sufficiently large for training good representations. These so-called "general domain" corpora are today the basis on which most word embeddings are trained, greatly limiting their use in more specific areas. In fact, "specialized domains" like the medical domain usually manifest enough lexical, semantic and stylistic idiosyncrasies (e.g. use of acronyms and technical terms) that general-purpose word embeddings are unable to effectively encode out-of-the-box. In this thesis, we explore how different kinds of resources may be leveraged to train domain-specific representations or further specialise preexisting ones.

Specifically, we first investigate how in-domain corpora can be used for this purpose. In particular, we show that both corpus size and domain similarity play an important role in this process and propose a way to leverage a small corpus from the target domain to achieve improved results in low-resource settings. Then, we address the case of BERT-like models and observe that the general-domain vocabularies of these models may not be suited for specialized domains. However, we show evidence that models trained using such vocabularies can be on par with fully specialized systems using in-domain vocabularies—which leads us to accept re-training general domain models as an effective approach for constructing domain-specific systems. We also propose CharacterBERT, a variant of BERT that is able to produce word-level open-vocabulary representations by consulting a word's characters. We show evidence that this architecture leads to improved performance in the medical domain while being more robust to misspellings.

Finally, we investigate how external resources in the form of knowledge bases may be leveraged to specialise existing representations. In this context, we propose a simple approach that consists in constructing dense representations of these knowledge bases then combining these knowledge vectors with the target word embeddings. We generalise this approach and propose Knowledge Injection Modules, small neural layers that incorporate external representations into the hidden states of a Transformer-based model. Overall, we show that these approaches can lead to improved results, however, we intuit that this final performance ultimately depends on whether the knowledge that is relevant to the target task is available in the input resource.

All in all, our work shows evidence that both in-domain corpora and knowledge may be used to construct better word embeddings for specialized domains. In order to facilitate future research on similar topics, we open-source our code and share pretrained models whenever appropriate.

Université Paris-Saclay
CNRS, LISN, Campus universitaire bât 507
Rue du Belvedère / 91405 Orsay CEDEX